Structured prediction

L101: Machine Learning for Language Processing

Andreas Vlachos

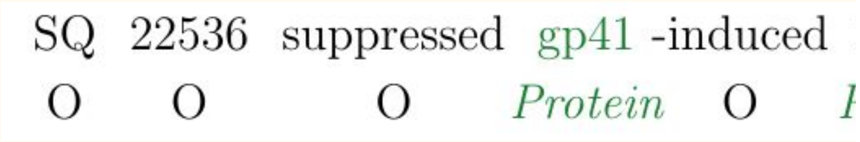
Structured prediction in NLP?

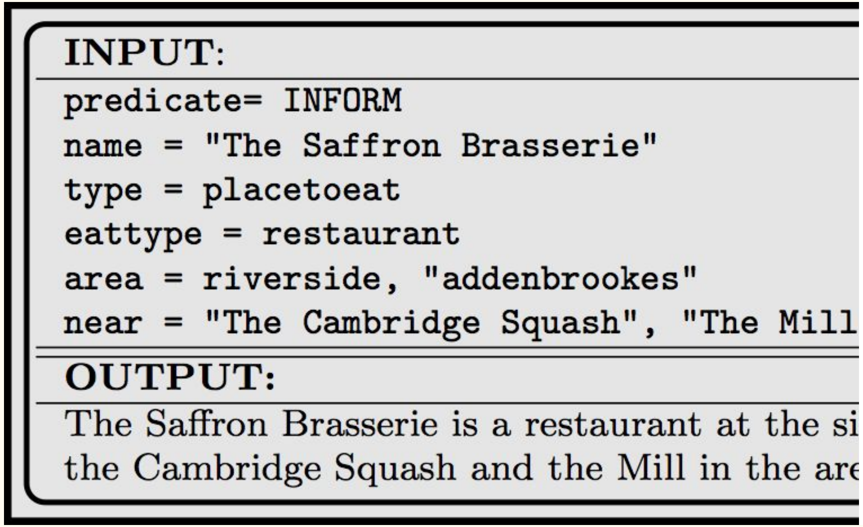
Given a piece of text, assign a *structured output*, typically a structure consisting of discrete labels

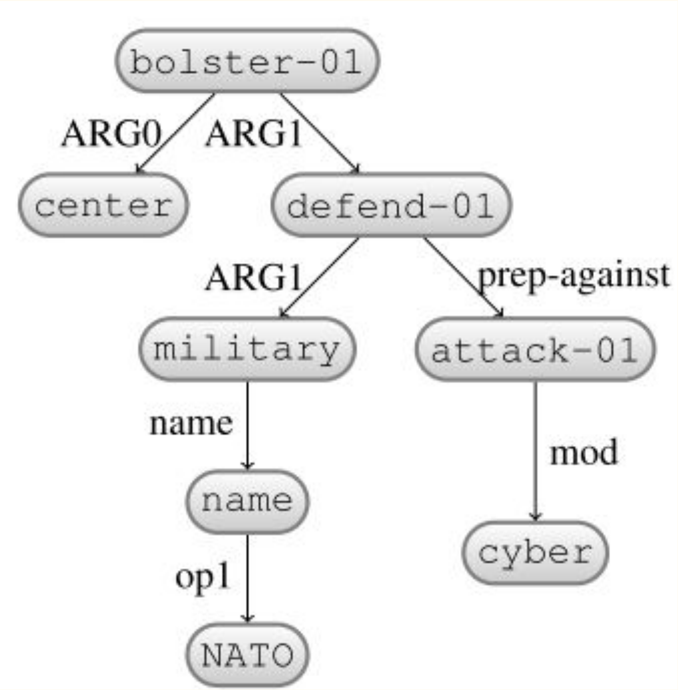
What could a structured output be?

- Sequence of part of speech tags
- Syntax tree
- SQL query
- Set of labels (a.ka. multi-label classification)
- Sequence of words (wait for the next lecture)
- etc.

Structured prediction in NLP is everywhere







Sequences of labels, words and graphs combining them

Structured prediction definition

Given an input ***x*** (e.g. a sentence) predict ***y*** (e.g. a PoS tag sequence):

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} score(x, y)$$

Where ***Y*** is rather large and often depends on the input (e.g. $\boldsymbol{L}^{|x|}$ in PoS tagging)

Is this large-scale classification?

- Yes, but with many, many classes
- Yes, but with classes not fixed in advance
- Yes, but with dependencies between parts of the output

Depending on how much the difference is, you might want to just classify

Structured prediction variants

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} score(x, y)$$

Linear models (structured perceptron)

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \Phi(x, y)$$

Generative models (HMMs)

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(x, y) = \arg\max_{y \in \mathcal{Y}} P(x|y)P(y)$$

Discriminative probabilistic models (conditional random fields)

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y|x)$$

Most of the above can use both linear and non-linear features, e.g. CRF-LSTMs

Structured perceptron

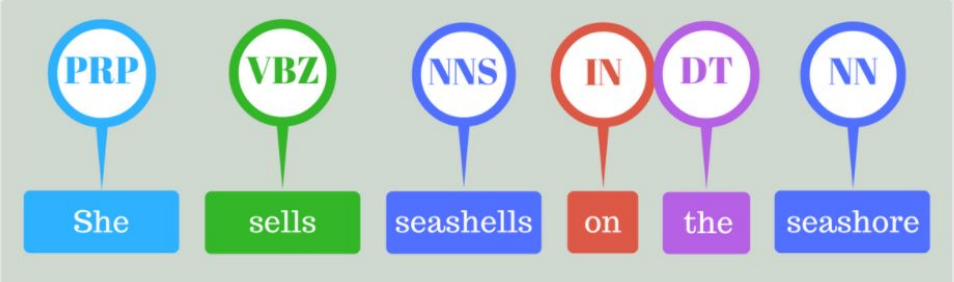
$$\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \Phi(x, y)$$

We need to learn ***w*** from training data

$$D = \{(x^1, y^1), \ldots (x^M, y^M)\}$$

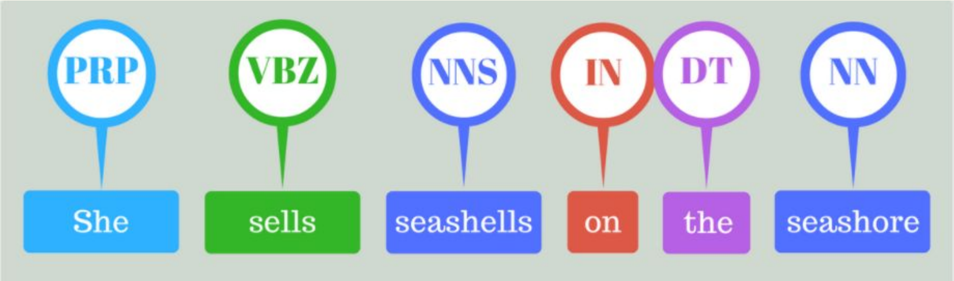
And define a joint feature map Φ***(x,y)***.

Ideas for PoS tagging?



Structured perceptron features

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \Phi(x, y)$$



Two kinds of features:

- Features describing dependencies in the output (without these: classification)
- Features describing the match of the input to the output

Feature factorization, e.g. adjacent labels:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \sum_i \phi(x, i, y_i, y_{i-1})$$

Does this restrict our modelling flexibility?

Perceptron training (reminder)

Input: training examples $\mathcal{D} = \{(x^1, y^1), \ldots (x^M, y^M)\}$
Initialize weights $w = (0, ..., 0)$
for $(x, y) \in \mathcal{D}$ **do**
  Predict label $\hat{y} = sign(w \cdot \phi(x))$
  if $\hat{y} \neq y$ **then**
    Update $w = w + y\phi(x)$
  end if
end for

Learn compatibility between positive class and instance

Structured Perceptron training (Collins, 2002)

Input: training examples $\mathcal{D} = \{(x^1, y^1), \ldots (x^M, y^M)\}$

Initialize weights $w = (0, ..., 0)$

for $(x, y) \in \mathcal{D}$ **do**

  Predict label $\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \Phi(x, y)$ ← Decoding

  if $\hat{y} \neq y$ **then**

    Update $w = w + \Phi(x, y) - \Phi(x, \hat{y})$ ← Feature differences

  end if

end for

Compatibility between input and output

Feature factorization accelerates both decoding and feature updating

Averaging helps

Guess the features and weights (Xavier Carreras)

Training Data

- Maria is young

PER	-	-
Maria	is	young

- Athens is big

LOC	-	-
Athens	is	big

- Jack went to Athens

PER	-	-	LOC
Jack	went	to	Athens

- Argentina is bigger

LOC	-	-
Argentina	is	bigger

- Jack London went to South Pacific

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

- Argentina played against Chile

ORG	-	-	ORG
Argentina	played	against	Chile

Some answers

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Athens	is	big

PER	-	-	LOC
Jack	went	to	Athens

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \textsc{Lower},\text{-} \rangle} = +1$

~~$\mathbf{w}_{\langle \textsc{Upper},\textsc{per} \rangle} = +1$~~

$\mathbf{w}_{\langle \textsc{Upper},\textsc{loc} \rangle} = +1$

$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\textsf{Maria} \rangle} = +2$

$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\textsf{Jack} \rangle} = +2$

$\mathbf{w}_{\langle \textsc{NextW},\textsc{per},\textsf{went} \rangle} = +2$

$\mathbf{w}_{\langle \textsc{NextW},\textsc{org},\textsf{played} \rangle} = +2$

$\mathbf{w}_{\langle \textsc{PrevW},\textsc{org},\textsf{against} \rangle} = +2$

$\dots$

$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{per},\textsc{per} \rangle} = +100$

$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{loc},\textsc{loc} \rangle} = +100$

$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{loc},\textsc{per} \rangle} = -100$

$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{per},\textsc{loc} \rangle} = -100$

$\mathbf{w}_{\langle \textsc{NextW},\textsc{loc},\textsf{played} \rangle} = -1000$

Decoding

Assuming we have a trained model, decode/predict/solve the argmax/inference:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} score(x, y; \theta)$$

Isn't finding θ meant to be the slow part (training)?

Decoding is often necessary for training; you need to predict to update weights

Do you know a model where training is faster than decoding?

Hidden Markov Models! (especially if you don't do Viterbi)

Can be exact or inexact (to save computation)

Dynamic programming

If we have a factorized the scoring function, we can reuse the scores (**optimal substructure** property), e.g.: $\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \sum_i \phi(x, i, y_i, y_{i-1})$

Thus changing one part of the output, doesn't change all/most scores

Viterbi recurrence:

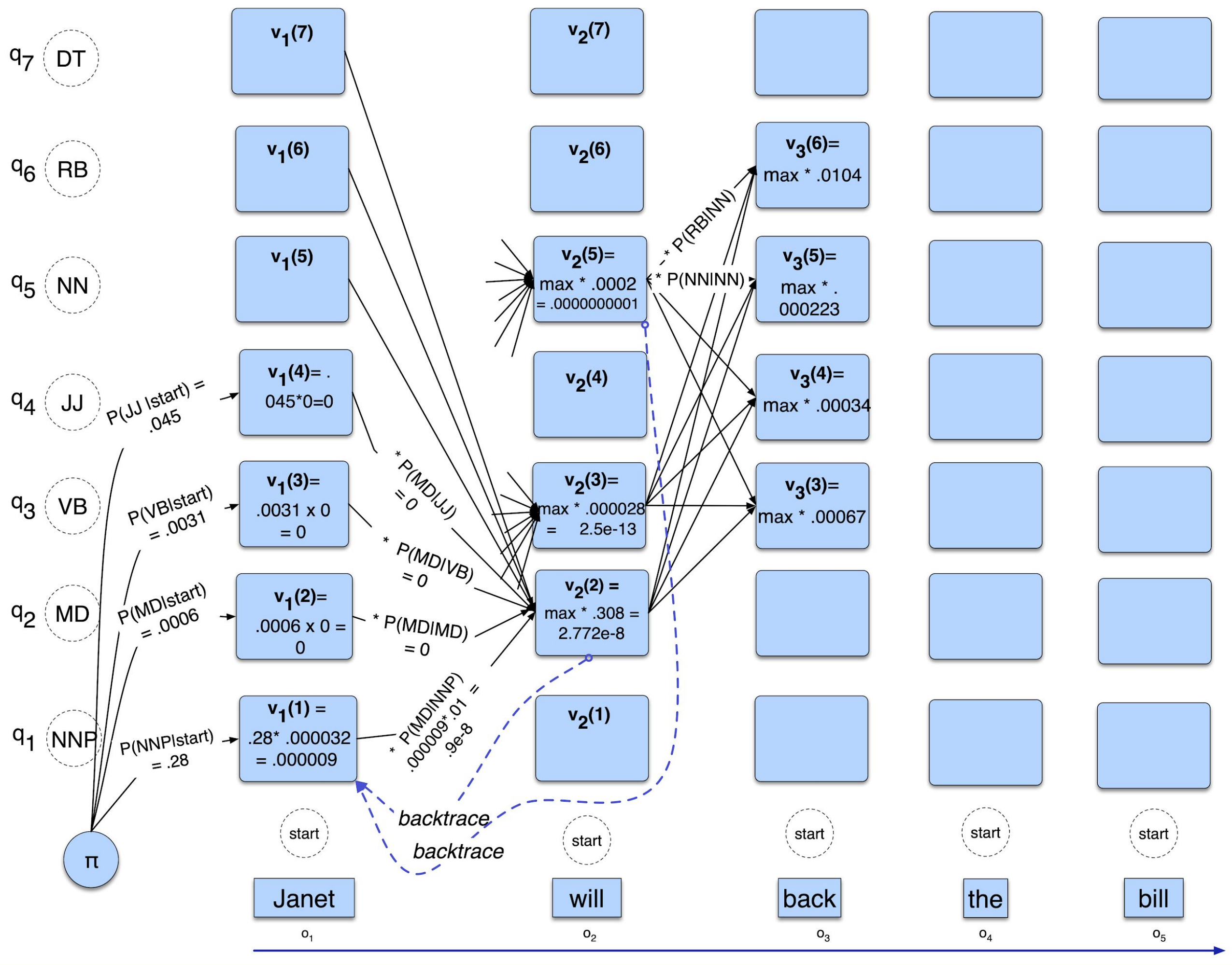
1. Assume we know for position $\boldsymbol{i}$ the best sequence ending with each possible $\boldsymbol{y_i}$
2. What is the best sequence up to position $\boldsymbol{i}+1$ for each possible $\boldsymbol{y_{i+1}}$?

An instance of shortest path finding in graphs

Viterbi in action

Apart from the best scores (max), need to keep pointers to backtrace to the labels (argmax)

Higher than first order Markov assumption is possible, but more expensive



[Jurafksy and Martin](#)

Conditional random fields

Multinomial logistic regression reminder:

$$P(\hat{y} = y) = \frac{\exp(w_y \cdot \phi(x))}{\sum_{y' \in \mathcal{Y}} \exp(w_{y'} \cdot \phi(x))}$$

Conditional random field is a giant of the same type (softmax and linear scoring):

$$P(\hat{y} = y | x; w) = \frac{\exp(w \cdot \Phi(x, y))}{\sum_{y' \in \mathcal{Y}^{|x|}} \exp(w \cdot \Phi(x, y'))}$$

The denominator is independent of y: needs to be calculated over all ys!

Often referred to as the partition function

Conditional random fields in practice

$$P(\hat{y} = y|x;w) = \frac{\exp(w \cdot \Phi(x,y))}{\sum_{y' \in \mathcal{Y}^{|x|}} \exp(w \cdot \Phi(x,y'))}$$

Factorize the scoring function:

$$w \cdot \Phi(x,y) = w \cdot \sum_i \phi(x, i, y_i, y_{i-1})$$

Dynamic programming to the rescue again: **forward-backward** algorithm

This allows us to train CRF by minimizing the convex negative log likelihood:

$$w^\star = \arg\min_w \sum_{(x,y) \in D} logP(y|x;w)$$

If you factorize the probability distribution: $P(\hat{y} = y|x;w) = \prod_{i=1}^{|x|} P(y_i|y_{i-1}, x; w)$

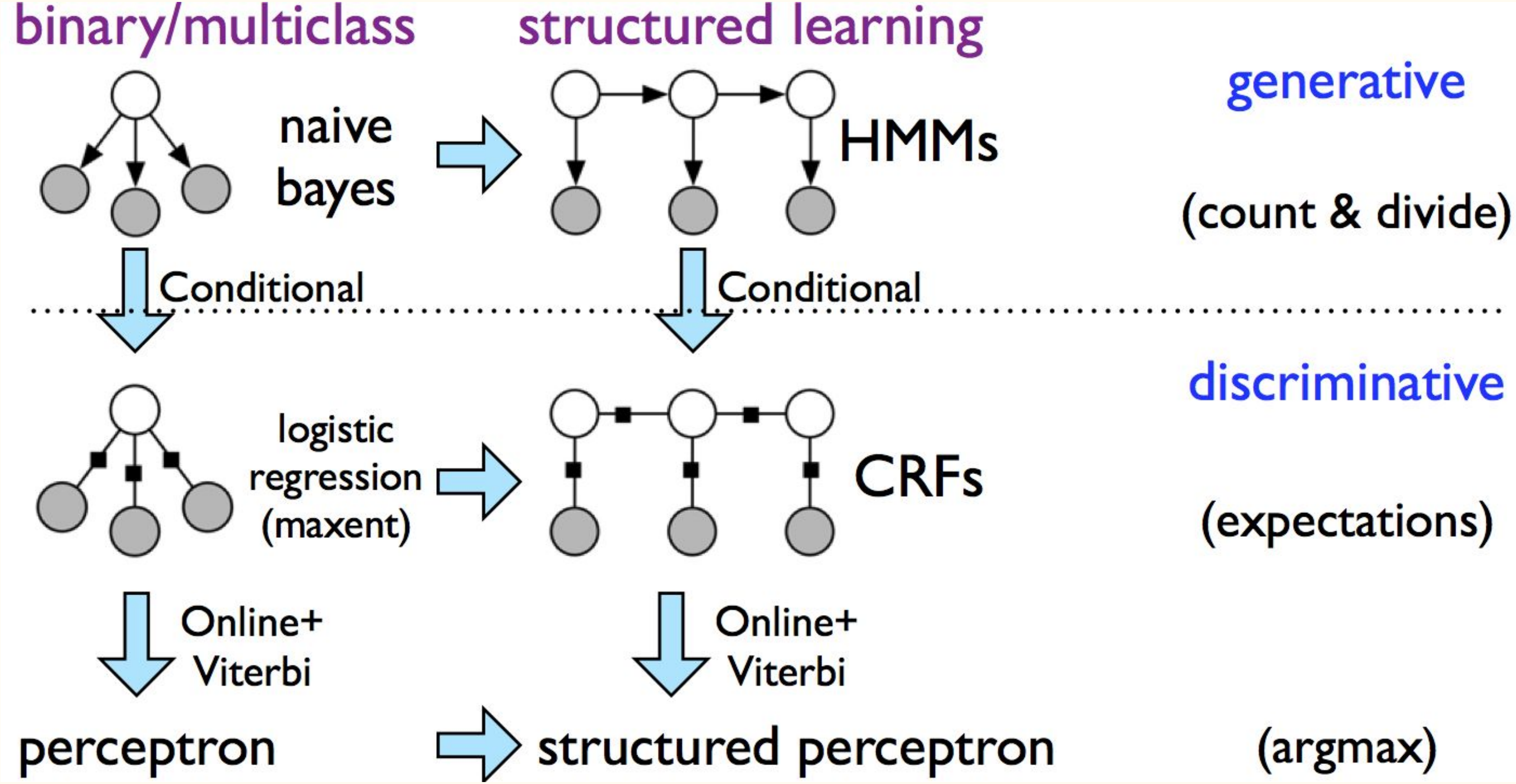
Maximum Entropy Markov Models: train logistic regression, Viterbi at inference

An overview

Xavier Carreras
AthNLP2019

	Feature flexibility: input-output representation	Decoding: exact prediction?
classification: label classifiers	only individual labels	yes
Lecture 7!: transition-based	full history of decisons	no (greedy, beam search)
Most common: factored	label factors	yes
Generalized binary classification: re-ranking	full	limited to active set

Another overview!



[Sutton and McCallum (2011)]

Things we didn't cover

Latent variable structured prediction:

- Intermediate labels for which we don't have annotation
- Can be thought of as hidden layers in NN (they are trained via "hallucinations")

Constrained inference:

- Sometimes you can prune your search space (remove invalid outputs)
- Reduces the crude enumeration outputs but can make inference slower when using dynamic programming (e.g. here on enforcing valid syntax trees)
- Dual decomposition is often considered: split it into two (simpler) constrained inference problems and solve them to agreement

Bibliography

- Kai Zhao's survey: very useful for structured perceptron
- Noah Smith's book: good overview
- Sutton and McCallum (2011): everything you wanted to know about conditional random fields
- Xavier Carreras's AthNLP2019 slides and video
- Michael Collins's notes on HMMs and Viterbi
- A blog post on implementing Viterbi and CRFs on pytorch