CRF example — written after class 10/20/16
(CS 585, UMass Amherst, Brendan O'Connor)

		finna	bless	us
y =	[S]	V	V	V

Tags: "V"erb and pr"O"noun (and [S]tart)
Let's use three feature templates:

Transition features: for example $f_{VV}(x,y)$ = number of V-V transitions in y

Word-tag observation features: for example $f_{V,dog}(x,y)$ = number of tokens that are word "dog" under a Verb tag

"ends with s"–tag features: $f_{V\text{-}s}(x,y)$ = number of tokens that end with -s and are tagged as Verb

(Global features have to be COUNTS: the reason why is further below.)
For 3 word vocabulary and 2 tag types, that's J=14 total features.
Assume we have fixed model weights θ and would like to score the goodness of the above tag sequence.

Global feature vector f(x,y) =

fSV	fSO	fVV	fVO	fOV	fOO	fV,finna	fV,bless	fV,us	fO,finna	fO,bless	fO,us	fV,-s	fO,-s
1	0	2	0	0	0	1	0	1	1	0	0	2	0

Model parameters θ =

θSV	θSO	θVV	θVO	θOV	θOO	θV,finna	θV,bless	θV,us	θO,finna	θO,bless	θO,us	θV,-s	θO,-s
-0.2	-0.8	+0.1	+0.5	+4.3	-0.3	-1.2	-0.1	+0.1	+5.3	-4.1	-0.3	+1.1	+2.2

Goodness score G(y) $= \theta' f(x,y) = \sum_{j=1}^{J} \theta_j f_j(x,y)$

= –0.2 + 0 + 0.2 +0 +0 +0 –1.2 + 0 +0.1 +5.3 +0 +0 +2.2 + 0

Global feature vector is from the sum of local feature vectors

$$f(x,y) = \sum_t f_t(y_{t-1}, y_t, x_t)$$

$f_t(y_{t-1}, y_t, x_t)$ = local feature vector including the transition between these two tags, and the observation of word at position t.

The local features are, for example:

f_{VV}(yprev, ycur, curword) = {1 if yprev=V and ycur=V, else 0}

$f_{V,dog}$(yprev, ycur, curword) = {1 if ycur=V and curword="dog", else 0}

$f_{V,-s}$(yprev, ycur, curword) = {1 if ycur=V and curword ends in "s", else 0}

And so on, repeated for different tags and words.

Example

	trans. feats						obs. feats							
	fSV	fSO	fVV	fVO	fOV	fOO	fV,finna	fV,bless	fV,us	fO,finna	fO,bless	fO,us	fV,-s	fO,-s
f(START, V, finna)	1	0	0	0	0	0	1	0	0	0	0	0	0	0
+ f(V, V, bless)	0	0	1	0	0	0	0	1	0	0	0	0	1	0
+ f(V, V, us)	0	0	1	0	0	0	0	0	1	0	0	0	1	0
= f(x=finna bless us, y=V V V) =	1	0	2	0	0	0	1	0	1	1	0	0	2	0

Local feature decomposition implies that the scoring function decomposes, too.

$$G(y) = \theta' f(x,y) = \theta' \sum_t f_t(y_{t-1}, y_t, x_t) = \sum_t \theta' f_t(y_{t-1}, y_t, x_t)$$

= θ'f(START, V, finna) + θ'f(V, V, bless) + θ'f(V, V, us)

= dotprod (| -0.2 | -0.8 | +0.1 | +0.5 | +4.3 | -0.3 | -1.2 | -0.1 | +0.1 | +5.3 | -4.1 | -0.3 | +1.1 | +2.2 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |)

+ dotprod (| -0.2 | -0.8 | +0.1 | +0.5 | +4.3 | -0.3 | -1.2 | -0.1 | +0.1 | +5.3 | -4.1 | -0.3 | +1.1 | +2.2 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |)

+ dotprod (| -0.2 | -0.8 | +0.1 | +0.5 | +4.3 | -0.3 | -1.2 | -0.1 | +0.1 | +5.3 | -4.1 | -0.3 | +1.1 | +2.2 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |)

In the assignment we're compiling these local feature scoring functions into A and B_t functions.

A(yprev, ycur) is the matrix of transition logprobs.
(If this was an HMM, A would be the able of log condprobs for possible transitions.)

$$A(y_{t-1}, y_t) = \sum_{j \in transfeats} \theta_j f_j(y_{t-1}, y_t, x_t)$$

B_t(ycur) is the log-prob weight for tag ycur, according to observation information at t.
(If this was an HMM, Bt would be the column of log condprobs for the word.)

$$B_t(y_t) = \sum_{j \in obsfeats} \theta_j f_j(null, y_t, x_t)$$

I'm writing "null" for the prev tag because the obs features don't depend on the prev tag: only the current one.

IIn HW4, note we separate the local feature function into two feature functions: one for the transition feature, and one for observation features. In lecture I tried to talk a single local feature function combining transition and observation features at the position just because that's how the Chen blogpost does it.

Why'd we remove the *x*'s from the math notation? We could've kept them if we wanted to. We just want to set up the graph weights for the Viterbi algorithm. Viterbi doesn't need to know the words. It only has to know the log-prob weights for different tags at each timestep, and the log-prob weights for different tag transitions.

Sparse vectors: The feature vectors are big long vectors where most elements are 0. In practice, you should never actually allocate an array of length J in your computer's memory. Instead, use a "sparse representation" where you use a Python dict where you have key-value pairs only for features with a nonzero value.