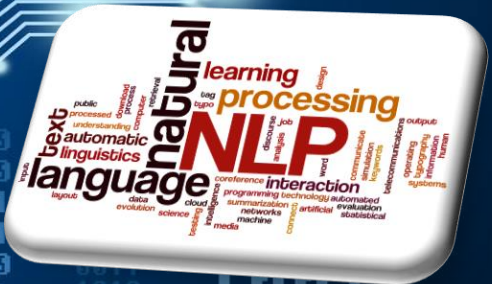
Cairo University

Cairo University
Faculty of Engineering
Computer Engineering Department

Natural Language Processing

Dr. Sandra Wahid



Conditional Random Field (CRF)

- While the HMM is a useful and powerful model, it turns out that HMMs need a number of augmentations to achieve high accuracy.
  - For example, in POS tagging as in other tasks, we often run into **unknown words**.
  - It would be great to have ways to add **arbitrary features** to help:
    - Capitalization or morphology (words starting with capital letters are likely to be proper nouns, words ending with -ed tend to be past tense)
    - Knowing the previous or following words might be a useful feature (if the previous word is *the*, the current tag is unlikely to be a verb).

- In general, it's hard for generative models like HMMs to add arbitrary features directly into the model in a clean way.
  - Logistic regression is a log-linear model for combining arbitrary features,

  → But logistic regression isn't a sequence model; it assigns a class to a single observation.

- There is a **discriminative sequence** model based on **log-linear** models: the conditional random field (CRF)→ we will describe **linear chain CRF**, the version of the CRF most commonly used for language processing, and the one whose conditioning closely matches the HMM.

Conditional Random Field (CRF)

We have a sequence of input words X = x1…xn and want to compute a sequence of output tags Y = y1…yn.

- In HMM:

$$\begin{aligned}\hat{Y} &= \underset{Y}{\operatorname{argmax}}\, p(Y|X)\\ &= \underset{Y}{\operatorname{argmax}}\, p(X|Y)p(Y)\\ &= \underset{Y}{\operatorname{argmax}} \prod_i p(x_i|y_i) \prod_i p(y_i|y_{i-1})\end{aligned}$$

- In CRF: we compute the **posterior p(Y|X)** directly:

$$\hat{Y} = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}}\, P(Y|X)$$

The CRF does not compute a probability for each tag at each time step.
Instead, at each time step the CRF computes log-linear functions over a set of relevant features.

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

The function F maps an entire input sequence X and an entire output sequence Y to a feature vector.

Let’s assume we have K features, with a weight w_k for each feature F_k

Conditional Random Field (CRF)

- Re-write as:

$$p(Y|X) = \frac{1}{Z(X)}\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)$$

$$Z(X) = \sum_{Y'\in\mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)$$

- These K functions $F_k(X,Y)$ are called **global features**

→since each one is a property of the entire input sequence X and output sequence Y.

- Computed by decomposing into a sum of **local features** for each position i in Y:

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

Each of these local features fk in a linear-chain CRF is allowed to make use of:

- current output token yi,
- previous output token yi-1,
- entire input string X (or any subpart of it),
- current position i.

what characterizes a linear chain CRF: this limitation makes it possible to use versions of the efficient Viterbi and Forward-Backwards algorithms from the HMM.

A general CRF, by contrast, allows a feature to make use of any output token.

Features in a CRF POS Tagger

- Some legal features representing common situations might be the following:

$$\mathbb{1}\{x_i = the,\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN},\ x_{i+1} = Street,\ y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB},\ y_{i-1} = \text{AUX}\}$$

Proper noun: name of a person, organization, place, etc.

For simplicity, we'll assume all CRF features take on the value 1 or 0.

- Specific features can be automatically populated by using **feature templates:**
  - These templates automatically populate the set of features from every instance in the training and test set.

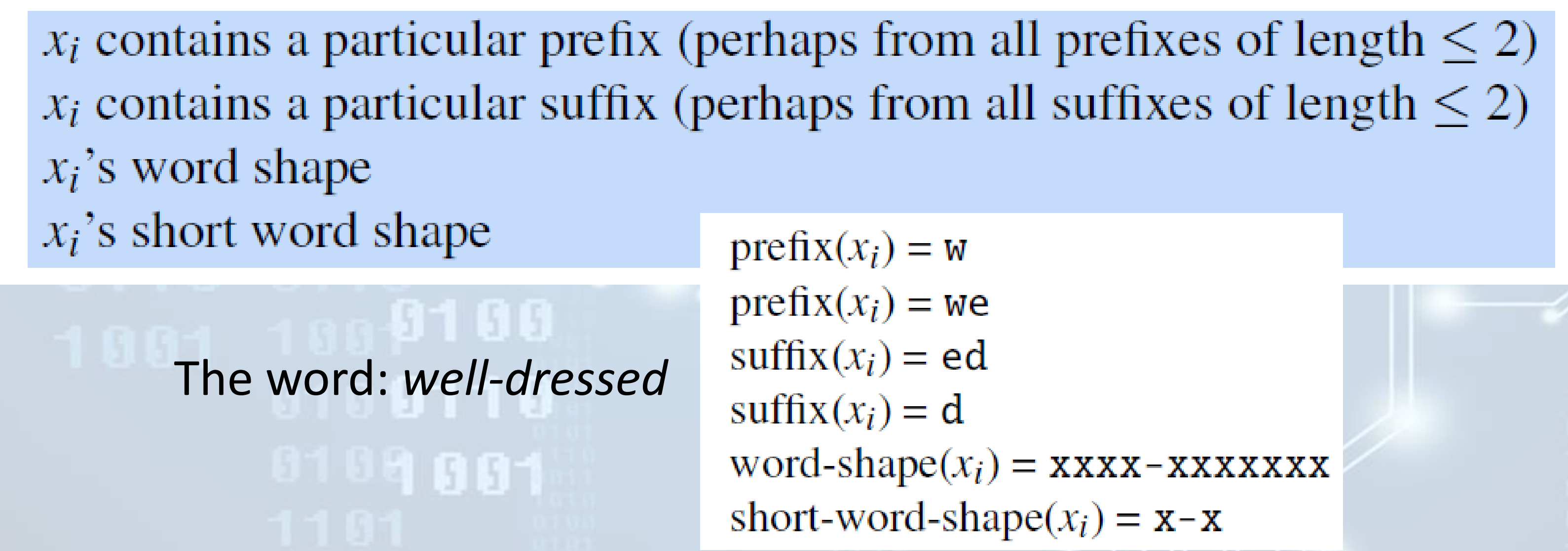
$$\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$$

example: Janet/NNP will/MD back/VB the/DT bill/NN, when xi is the word back, the following features would be generated and have the value 1 (we've assigned them arbitrary feature numbers):

$$f_{3743}\text{: } y_i = \text{VB and } x_i = \text{back}$$
$$f_{156}\text{: } y_i = \text{VB and } y_{i-1} = \text{MD}$$
$$f_{99732}\text{: } y_i = \text{VB and } x_{i-1} = \text{will and } x_{i+2} = \text{bill}$$

Features in a CRF POS Tagger

- It's also important to have features that help with **unknown words:**
  - **word shape features:**
    - which represent the abstract letter pattern of the word by mapping: lower-case letters to 'x', upper-case to 'X', numbers to 'd', retaining punctuation.
    - Examples: I.M.F would map to X.X.X. and DC10-30 would map to XXdd-dd.
  - **short word shape features:**
    - consecutive character types are removed, so words in all caps map to X, words with initial-caps map to Xx,
    - Examples: DC10-30 would be mapped to Xd-d but I.M.F would still map to X.X.X.
  - Example features:



x_i contains a particular prefix (perhaps from all prefixes of length ≤ 2)

x_i contains a particular suffix (perhaps from all suffixes of length ≤ 2)

x_i's word shape

x_i's short word shape

The word: *well-dressed*

$\text{prefix}(x_i) = \texttt{w}$

$\text{prefix}(x_i) = \texttt{we}$

$\text{suffix}(x_i) = \texttt{ed}$

$\text{suffix}(x_i) = \texttt{d}$

$\text{word-shape}(x_i) = \texttt{xxxx-xxxxxxx}$

$\text{short-word-shape}(x_i) = \texttt{x-x}$

Features in a CRF POS Tagger

- The known-word templates are computed for every word seen in the training set.
- The unknown word features can also be computed for all words in training, or only on training words whose frequency is below some threshold.

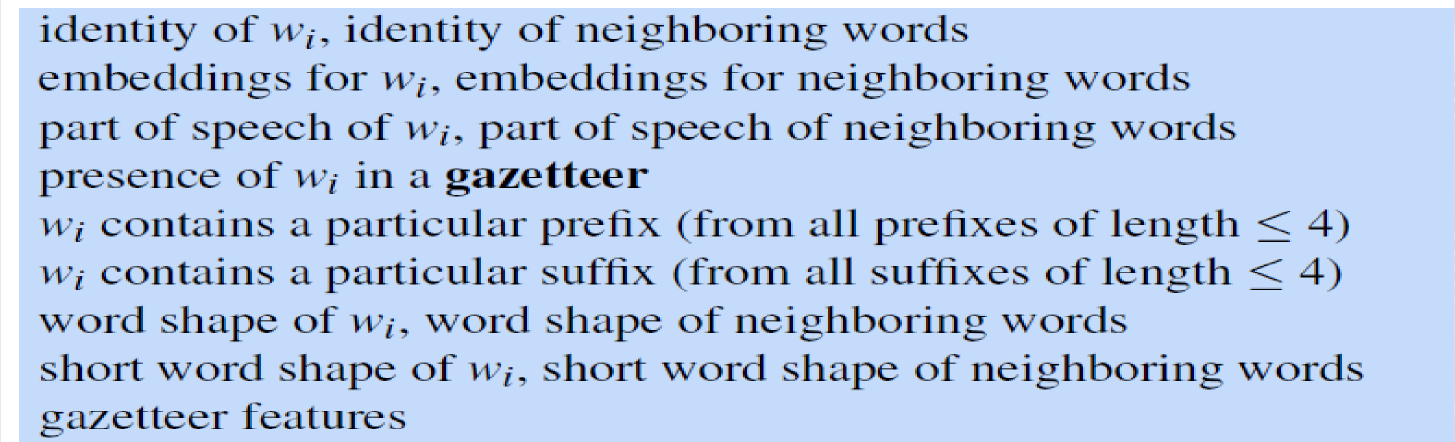
→The result is a very large set of features:

Generally, a **feature cutoff** is used in which features are thrown out if they have count < 5 in the training set.

- In a CRF we don’t learn weights for each of these local features fk.
- Instead, we first sum the values of each local feature (for example feature f3743) over the entire sentence, to create each global feature (for example F3743).
- It is those global features that will then be multiplied by weight w3743.
- Thus, for training and inference there is always a **fixed set of K features with K weights**, even though the length of each sentence is different.

Features for CRF Named Entity Recognizers

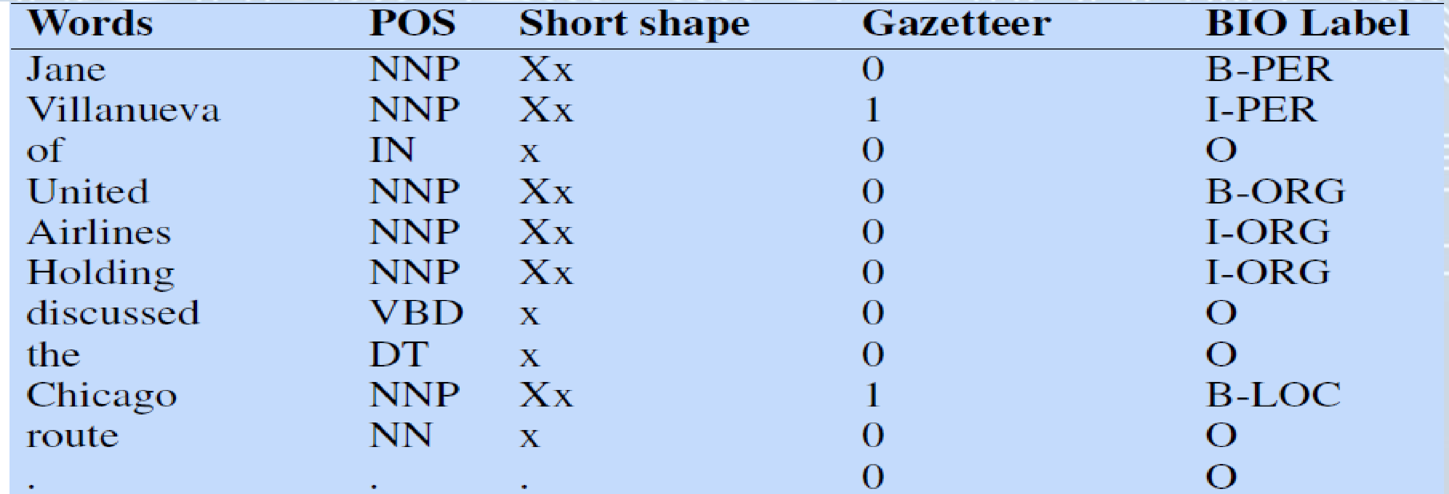
- A CRF for NER makes use of very similar features to a POS tagger:



identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
gazetteer features

  - **gazetteer:** list of place names, often providing millions of entries for locations with detailed geographical and political information.
  - name-lists, lists of corporations or products, …etc.

- Some NER features for a sample sentence:



Words	**POS**	**Short shape**	**Gazetteer**	**BIO Label**
Jane	NNP	Xx	0	B-PER
Villanueva	NNP	Xx	1	I-PER
of	IN	x	0	O
United	NNP	Xx	0	B-ORG
Airlines	NNP	Xx	0	I-ORG
Holding	NNP	Xx	0	I-ORG
discussed	VBD	x	0	O
the	DT	x	0	O
Chicago	NNP	Xx	1	B-LOC
route	NN	x	0	O
.	.	.	0	O

Inference and Training for CRFs

$$
\begin{aligned}
\hat{Y} &= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}}\, P(Y|X) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}}\, \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}}\, \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)
\end{aligned}
$$

Inference: How do we find the best tag sequence $\hat{Y}$ for a given input X?

we can ignore the exp function and the denominator Z(X) because this doesn't change the argmax, and the denominator Z(X) is constant for a given observation sequence X.

replacing transition and emission probabilities with the CRF features

- Using Viterbi Algorithm:

$$
v_t(j) = \max_{i=1}^{N}\; v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t) \quad 1 \leq j \leq N, 1 < t \leq T
$$

Training: given a sequence of observations, feature functions, and corresponding outputs, we use stochastic gradient descent to train the weights to maximize the log-likelihood of the training corpus.

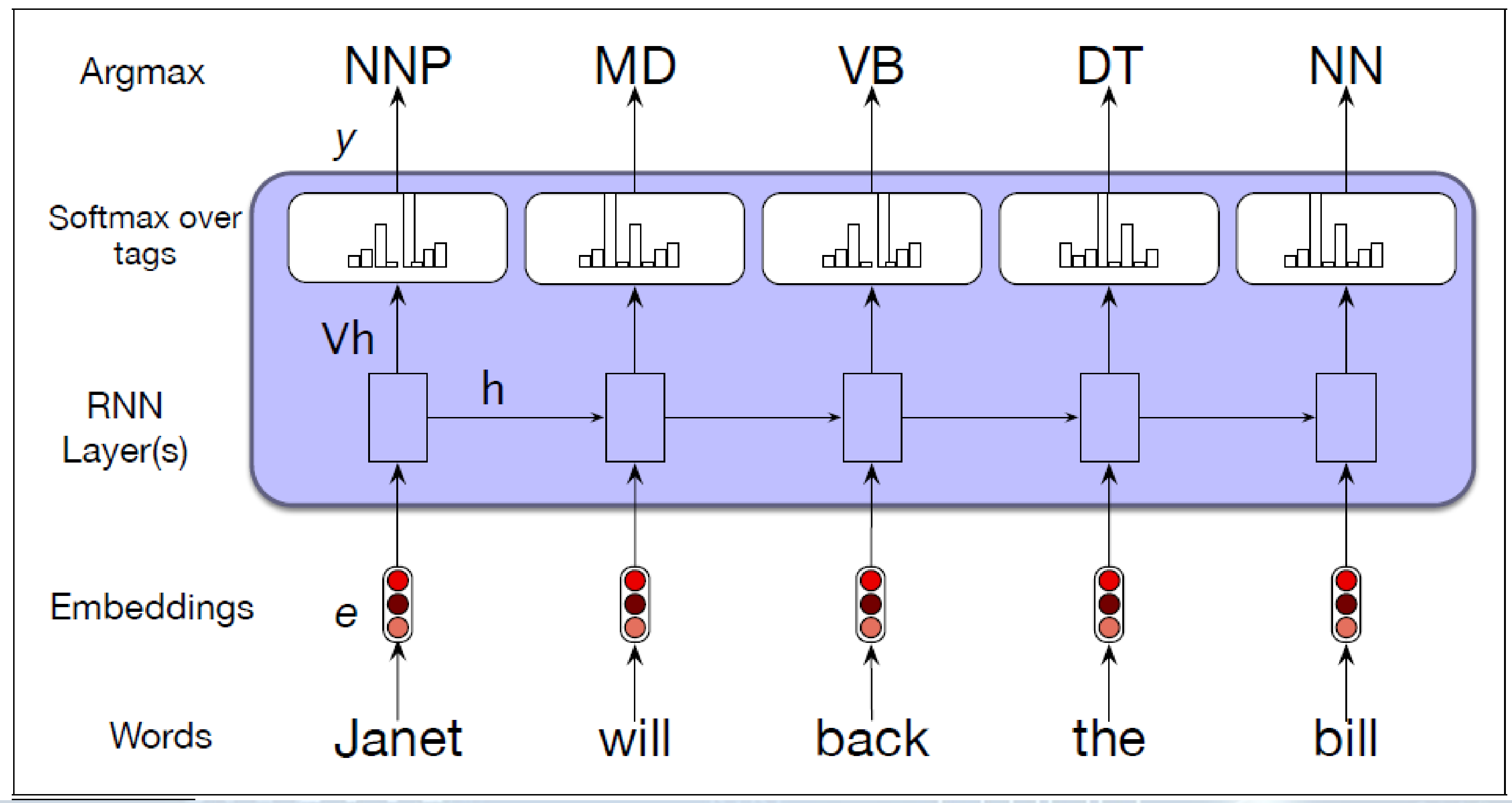
Evaluation

- Part-of-speech taggers are evaluated by the standard metric of **accuracy**.

- Named entity recognizers are evaluated by **recall**, **precision**, and **F1 measure**.
  - The fact that named entity tagging has a segmentation component which is not present in tasks like text categorization or part-of-speech tagging causes some problems with evaluation.
  - For example, a system that labeled *Jane* but not *Jane Villanueva* as a person would cause two errors:
    - a false positive for O
    - a false negative for I-PER

RNN Sequence Labeling

- Inputs: are word embeddings
- Outputs: are tag probabilities generated by a softmax layer over the given tagset.



0100
0110
1001
0100
0110
1001
0101
0100
1101
0100
0110
1001
1010
0110
1001
0101
0100
1101
0101
1110
1010
0110
1001
0101
0100
0110
1001

Thank You