NER – To predict a sequence of labels from a sequence of input Xs.

Probabilistic Graphical Models are used to model this problem.

CRF is a discriminative undirected graphical probabilistic model. This is used for modeling NER.

<u>Undirected Graphical Model</u>: set of all distributions that can be written in the form –

$$P(x,y) = \frac{1}{Z} \prod_{A} \Psi_A(x_A, y_A)$$

for any choice of factors $f = \{\Psi_A\}$. A is a subset of V, $V = X \cup Y$

$$Z = \sum_{x,y} \prod_{A} \Psi_A(x_A, y_A)$$

Z is a Normalization factor also called as Partition function.

(Computing Z is intractable but there are methods to approximate it)

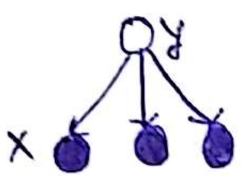
Model refers to the family of distributions whereas random field is used to refer to a single distribution.

<u>Application of Graphical Models</u>:

1) <u>Classification</u>: → Naive Bayes classifier → Logistic Regression

Naive Bayes classifier:

$$P(y,x) = P(y) \times \prod_{K=1}^{K} P(x_K/y)$$



Joint Distribution

X → independent

Logistic Regression:

$$P(y/x) = \frac{1}{Z(x)} \exp\left\{\lambda_y + \sum_{J=1}^{K} \lambda_{y,J} x_J\right\}$$

conditional distribution.

X → No need of independence

(1)

Scanned by CamScanner

2) <u>Sequence Models</u>: Ability to model many variables that are interdependent. In NER: we could classify each word independently as one of either Person, Location, Organisation, other. The problem is that the Named entities are not independent. They are dependent on each other, hence we need sequence models.

HMM (Hidden Markov Models) relax the independence assumption or incorporate the interdepence b/w entities by arranging the output variables in Linear chain.

To model the Joint Probability distribution $P(x,y)$, HMM makes 2 assumptions:

1. Each state depends only on its immediate predessor
   y_t depends on y_{t-1} only and is independent of y_{t-2} etc.
2. Each observation x_t depends only on current state y_t

<u>HMM</u>: $y \rightarrow$ state sequence
$x \rightarrow$ observation sequence

$$P(y,x) = \prod_{t=1}^{T} P(y_t / y_{t-1}) \, P\left(\frac{x_t}{y_t}\right)$$

<u>Discriminative & Generative Models</u>:

Naive Bayes classifier $\rightarrow$ Generative
Logistic Regression $\rightarrow$ Discriminative

<u>Generative Models</u> $\rightarrow$ models Joint Probability distribution $P(x,y)$. We need $P(x)$ also here and its difficult to get $P(x)$ if X_s are dependent.

<u>Discriminative Models</u> $\rightarrow$ models conditional probability. Don't include a model of $P(x)$, which is not needed for classification anyway.

Scanned by CamScanner

Due to independence assumption HMM relies on only one feature (word's identity) for NER.

What happens if we include dependent features in Generative Models (eg Naive Bayes)?

- Imagine a feature set X $(x_1, x_1, x_2, x_2, \cdots x_n, x_n)$ with all repeated features. This will increase the confidence of naive bayes probability estimates. even though no new information has been added. Since the probability estimates are poor, we can't use it in sequence models as inference combines evidence from different parts of model.

Naive Bayes Vs. Logistic Regression (Generative - Discriminate interchange)

Naive Bayes & Logistic Regression are same except that former is generative & latter is discriminative rest everything is same b/w the two.

Naive Bayes model defines the same family of distributions as the logistic regression if we interpret it generatively as

$$P(y,x) = \frac{\exp\{\sum_k \lambda_k f_k(y,x)\}}{\sum_{\bar{y}\,\bar{x}} \exp\{\sum_k \lambda_k f_k(\bar{y},\bar{x})\}}$$

That is if Naive Bayes is trained to maximize the conditional likelihood, we recover the same classifier as logistic Regression. Conversely if LR is trained to maximize the Joint likelihood $P(y,x)$ then we recover the same classifier as Naive Bayes.

Scanned by CamScanner

Linear chain CRF : CRF is combination of Discriminative & sequence modeling. CRF is a type of discriminative, undirected graphical model.

Linear chain CRF is a special type of CRF that assumes the current state depends only on the previous state. We define linear chain CRF motivating them from HMMs.

We begin by considering the conditional distribution $P(y/x)$ that follows from the joint distribution $P(y,x)$ of an HMM. The conditional distribution is in fact a conditional random field with a particular choice of feature functions.

We can rewrite HMM as $\rightarrow$ $P(y,x) = \frac{1}{Z} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\}$

$$P(y,x) = \prod_{t=1}^{T} P\left(\frac{y_t}{y_{t-1}}\right) P\left(\frac{x_t}{y_t}\right)$$

Now we write condition distribution $P(y/x)$ from the HMM

$$P(y/x) = \frac{P(y,x)}{\sum_{y'} P(y',x)} = \frac{\exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\}}{\sum_{y'} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y'_t, y'_{t-1}, x_t) \right\}}$$

Thus linear chain CRF is a distribution $P(y/x)$ -

$$P(y/x) = \frac{1}{Z(x)} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\},$$

$$Z(x) = \sum_{y} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\}$$

Scanned by CamScanner

Thus, If Joint $P(y,x)$ factorizes as an HMM, then the associated conditional distribution $P(y/x)$ is a linear chain CRF.

Parameter Estimation of Linear chain CRF:

estimate $\theta = \{\lambda_k\}$

Training Data $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$

$x^{(i)} = \{x_1^{(i)}, x_2^{(i)} \ldots x_T^{(i)}\}$

$y^{(i)} = \{y_1^{(i)}, y_2^{(i)} \ldots y_T^{(i)}\}$

N = # of examples

t = # of terms/tokens in each example.

Parameters are estimated by penalized maximum likelihood.

$$l(\theta) = \sum_{i=1}^{N} \log P(y^{(i)}/x^{(i)})$$

$$l(\theta) = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N} \log Z(x^{(i)})$$

Adding Regularization term $\longrightarrow - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2}$

$l(\theta)$ can't be maximized in closed form, so numerical optimization is used.

$$\frac{\partial l}{\partial \lambda_k} = \sum_{i=1}^{N}\sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'} f_k(y, y', x_t^{(i)}) P(y, y'/x^{(i)}) - \sum_{k=1}^{K} \frac{\lambda_k}{\sigma^2}$$

Scanned by CamScanner

$l(\theta)$ is concave, hence has exactly one global optimum.

Simplest approach to optimize l is steepest ascent along the gradient.

Newton's method converges faster because it takes into account curvature of likelihood, but requires computing Hessian (matrix of 2nd order derivatives). The size is quadratic in # of parameters. But we have millions/thousands of parameters so storing the full Hessian is not practical.

Current techniques for optimizing make approximate use of 2nd order information.

BFGS is a quasi-Newton method which has been successful. It computes an approximation to the Hessian from only the 1st derivative of the objective function. A full $K \times K$ approximation to Hessian still requires quadratic size.

A limited memory version of BFGS is used.

<u>Computational Cost of Training:</u>

Both the partition function $Z(x)$ in the likelihood and the marginal distributions $P(y_t, y_{t-1}/x)$ in the gradient can be computed by forward-backward which uses complexity of $O(TM^2)$.

Scanned by CamScanner

Considering all training instances each having different partition function and marginals total training cost will be $O(TM^2NG)$

N = # of training examples

G = # of Gradient computations by optimization procedure.

On a standard Named entity data set with 11 Labels and 200K words of training data, CRF took 2 hours.

On a POS tagging Dataset with 45 labels and 1 million words of training data, CRF training requires over a week.

Inference

There are 2 common inference problems for CRF

1) During training, computing the gradient requires marginal distributions for each edge $P(y_t, y_{t-1}/x)$ and computing the likelihood requires $Z(x)$.

(Forward-Backward is used for this)

2) Once we have estimated the parameters & have ~~[illegible]~~ the CRF $P(y/x)$, we need to label the sequence of X_s. ie, input words. ie, get the output labels.

Scanned by CamScanner

we compute the most likely labelling
$y^* = \arg\max_y P(y/x)$. Viterbi Algorithm
is used for this.

Hence, both inference can be solved by
dynamic programming - 1st by forward-backward
& 2nd by Viterbi Algorithm.

Scanned by CamScanner