NATURAL LANGUAGE PROCESSING

Chunking using CRF Model for Telugu Data

Srujai Varikuti (201101023) K.Sumant(201102180)

Abstract: Conditional Random Fields (CRFs) is a framework for building probabilistic models to segment and label sequence data. CRFs offer several advantages over hidden Markov models (HMMs) and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Here we have implemented chunking using various templates for CRF. The best template in the CRF chunking gave about 87.73% accuracy.

Keywords: Conditional Random Fields, Chunking, Language Models, Accuracy.

Introduction:

Chunking or shallow parsing is the task of identifying and segmenting the text into syntactically correlated word groups like noun phrase, verb phrase etc. It is considered as an intermediate step towards full parsing. This projects presents the exploitation of CRFs for Chunking of Telugu language. A rule based chunker is also used for chunking the data where the rules are written using the train data. Then we compare the results of each chunker.

Conditional Random Fields:

Conditional random fields (CRFs) are undirected graphical models developed for labeling sequence data. CRFs directly model $p(x \mid z)$, the *conditional* distribution over the hidden variables x given observations z. This model is different from generative models such as Hidden Markov Models or Markov Random Fields, which apply Bayes rule to infer hidden states. CRFs can handle arbitrary dependencies between the observations z, which gives them substantial flexibility in using high-dimensional feature vectors.

The nodes in a CRF represent hidden states, denoted $x = < x_1, x_2, \ldots, x_n >$, and data, denoted z. The nodes x_i, along with the connectivity structure represented by the undirected edges between them, define the conditional distribution $p(x \mid z)$ over the hidden states x. Let C be the set of cliques (fully connected subsets) in the graph of a CRF. Then, a CRF factorizes the conditional distribution into a product of *clique potentials* $\phi_c(z, x_c)$, where every c ε C is a clique in the graph and z and x_c are the observed data and the hidden nodes in the clique c, respectively. Clique potentials are functions that map variable configurations to non-negative numbers. Intuitively, a potential captures the "compatibility" among the variables in the clique: the larger the potential value, the more likely the configuration. Using clique potentials, the conditional distribution over hidden states is written as

$$p(x \mid z) = 1/Z(z)\left(\prod_{c \in C} \phi_c(z, x_c)\right) \qquad (1)$$

where $Z(z) = \sum \prod \phi_c(z, x_c)$ is the normalizing partition function.

$x\, c \in C$

The computation of this partition function can be exponential in the size of *x*. Hence, exact inference is possible for a limited class of CRF models only. Potentials $\phi_c(z, x_c)$ are described by log-linear combinations of *feature functions* f_c i.*e,*

$\phi_c(z, x_c) = \exp(w_c^T . f_c(z, x_c))$ --- (2) where w_c^T is called as a weight vector.

$f_c(z, x_c)$ is a function that extracts a vector of features from the variable values. Using feature functions, we rewrite the conditional distribution (1) as

$$P(x|z) = 1/Z(z) \exp\{ \sum_{c \in C} Wc^T . f_c(Z, X_c) \}$$

OUTPUTS:

Here we ran it for three different template files and given below are outputs and accuracies. These were calculated using the software CRF ++ which takes a train data and an feature file and generates a model file which will be used for testing test data. Following are the results of various feature files used in CRF++.

Experiment 1: Features: token, POS

Template 1

U00:%x[0,0]
U01:%x[0,1]

Results:

processed 163500 tokens with 109496 phrases; found: 129540 phrases; correct: 78970.
accuracy: **78.10%;** precision: **60.96%;** recall: **72.12%;** FB1: **66.07**

BLK: precision: 76.38%; recall: 68.90%; FB1: 72.44 6139
BLLK: precision: 0.00%; recall: 0.00%; FB1: 0.00 0
CCP: precision: 91.93%; recall: 92.19%; FB1: 92.06 1091
CP: precision: 0.00%; recall: 0.00%; FB1: 0.00 0
FRAGP: precision: 17.78%; recall: 4.44%; FB1: 7.11 90
JJP: precision: 14.43%; recall: 4.54%; FB1: 6.90 201
JPP: precision: 0.00%; recall: 0.00%; FB1: 0.00 9
NP: precision: 56.76%; recall: 70.08%; FB1: 62.73 78319
RBP: precision: 89.68%; recall: 90.17%; FB1: 89.93 3867
VG: precision: 0.00%; recall: 0.00%; FB1: 0.00 0
VGF: precision: 61.26%; recall: 91.51%; FB1: 73.39 34485
VGN: precision: 0.00%; recall: 0.00%; FB1: 0.00 1
VGNF: precision: 78.45%; recall: 43.36%; FB1: 55.85 4955
VGNFN: precision: 0.00%; recall: 0.00%; FB1: 0.00 0
VGNN: precision: 77.55%; recall: 23.39%; FB1: 35.93 383

Experiment 2: Features: token, POS of token, POS of previous token

Template 2

U00:%x[0,0]

U01:%x[0,1]

U02:%x[-1,1]

Results:

processed 163500 tokens with 109496 phrases; found: 116719 phrases; correct: 89265.

accuracy: **87.25%;** precision**: 76.48%;** recall: **81.52%;** FB1**: 78.92**

BLK: precision: 89.13%; recall: 88.04%; FB1: 88.58 6723

BLLK: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

CCP: precision: 96.69%; recall: 96.69%; FB1: 96.69 1088

CP: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

FRAGP: precision: 42.60%; recall: 51.94%; FB1: 46.81 439

JJP: precision: 30.74%; recall: 23.47%; FB1: 26.62 488

JPP: precision: 0.00%; recall: 0.00%; FB1: 0.00 9

NP: precision: 75.30%; recall: 82.21%; FB1: 78.60 69251

RBP: precision: 89.68%; recall: 91.32%; FB1: 90.49 3916

VG: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGF: precision: 75.90%; recall: 93.35%; FB1: 83.73 28395

VGN: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGNF: precision: 73.39%; recall: 48.77%; FB1: 58.60 5957

VGNFN: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGNN: precision: 66.45%; recall: 23.70%; FB1: 34.94 453

Experiment 3: Features: token, previous token, POS of token, POS of previous token

Template 3

U00:%x[0,0]

U01:%x[0,1]

U02:%x[-1,1]

U03:%x[-1,0]

Results:

processed 163500 tokens with 109496 phrases; found: 115913 phrases; correct: 90195.

accuracy: **87.73%**; precision: **77.81%;** recall: **82.37%;** FB1**: 80.03**

BLK: precision: 90.33%; recall: 87.14%; FB1: 88.71 6566

BLLK: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

CCP: precision: 96.32%; recall: 96.23%; FB1: 96.28 1087

CP: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

FRAGP: precision: 40.85%; recall: 53.33%; FB1: 46.27 470

JJP: precision: 36.85%; recall: 26.76%; FB1: 31.01 464

JPP: precision: 0.00%; recall: 0.00%; FB1: 0.00 1

NP: precision: 78.10%; recall: 84.33%; FB1: 81.10 68492

RBP: precision: 90.72%; recall: 91.26%; FB1: 90.99 3869

VG: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGF: precision: 75.29%; recall: 91.42%; FB1: 82.58 28029

VGN: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGNF: precision: 68.22%; recall: 50.85%; FB1: 58.27 6681

VGNFN: precision: 0.00%; recall: 0.00%; FB1: 0.00 0

VGNN: precision: 74.41%; recall: 14.88%; FB1: 24.80 254

OBSERVATIONS:

As we see here based on the feature template modifications the accuracy in our results changes. The state features and transition features in the feature file has a huge effect. If we take more features we are getting better results.

3) CONCLUSION:

The models designed for chunking based on CRF gave higher performance. The templates are very important in designing the CRF as they highly effect the performance of the chunking. CRF is best way as compared to Rule Based Model.

REFERENCES:

FOR CRF++:

http://crfpp.googlecode.com/svn/trunk/doc/index.html?source=navbar