EMNLP/VLC 2000

Japanese Dependency Structure Analysis Based on Support Vector Machines

Graduate School of Information Science,
Nara Institute of Science and Technology, JAPAN

Taku Kudoh, Yuji Matsumoto

{*taku-ku,matsu*}*@is.aist-nara.ac.jp*

Dependency Structure Analysis(1/3)

- Basic technique in Japanese sentence analysis
  - Analysis of relationship between phrasal units called "bunsetsu" ("chunks").
  - Each chunk modifies one of the right-side chunks
  - Dependencies do not cross each other
- 1. Building dependency matrix
     - Each element represents the probability of a dependency
  2. Finding the optimal combination of dependencies from the matrix
- Rule based approached → Corpus based statistical approach

Dependency Structure Analysis(2/3)

Problems of Conventional Frameworks(1/2)

- Must select “effective” features carefully
  - Trade off between over-fitting and over-generalization
  - The selection usually depends on heuristics

Problems of Conventional Frameworks(2/2)

- Difficulty in acquisition of an efficient combination of features
  - "Effective" selection of combinations usually decided by heuristics
  - The more specfic combinations we select, the larger computational overhead is required

Overview of the Talk

- Brief introduction to Support Vector Machines
  - How can SVMs cope with the problems of conventional frameworks?
- How do we apply SVMs to dependency analysis?
- Experiments and Evaluation
- Summary

Support Vector Machines (1/4)

- V.Vapnik 1995
- Two strong properties
  - High generalization performance independent of given feature dimension
  - Training with combinations (dependencies, co-occurrence) of more than one features without increasing computational overhead

Support Vector Machines (2/4)

- Separating positive and negative (binary) examples by **Linear Hyperplane**: $(\mathbf{w} \cdot \mathbf{x} + b, \quad \mathbf{w}, \mathbf{x} \in \mathbf{R^n}, b \in \mathbf{R})$
- Finding optimal hyperplane (parameter $\mathbf{w}, b$) with **Maximal Margin Strategy**

Support Vector Machines (3/4)

Two dashed lines (separating hyperplanes):

$$\mathbf{w} \cdot \mathbf{x} + b = \pm 1 \qquad \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}$$

Margin:

$$d = \frac{|\mathbf{w} \cdot \mathbf{x}_i + b - 1|}{\|\mathbf{w}\|} + \frac{|\mathbf{w} \cdot \mathbf{x}_i + b + 1|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Maximize Margin $d \leftrightarrow$ Minimize $\|\mathbf{w}\|$

Support Vector Machines (4/4)

Solving the following Optimizaion Problems:

$$\text{Minimize}: \quad L(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{Subject to}: \quad y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1 \ (i=1,\ldots,l)$$

Rewritten into dual form:

$$\text{Minimize}: \quad L(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j (\mathbf{x_i}\cdot\mathbf{x_j})$$

$$\text{Subject to}: \quad \alpha_i \geq 0,\ \sum_{i=1}^{l}\alpha_i y_i = 0 \qquad (i=1,\ldots,l)$$

Decision Function:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}\cdot\mathbf{x}+b) = \mathrm{sgn}(\sum_{i=1}^{l}\alpha_i y_i(\mathbf{x}_i\cdot\mathbf{x})+b)$$

Kernel Function (1/3)

The case we cannot separate the training data linearly

$$\Downarrow$$

Projecting training data onto a higher-dimensional space

$$\Phi(\mathbf{x}) : \{x_1,\ \ x_2\} \mapsto \{x_1,\ \ x_2,\ \ x_1 x_2\}$$

Kernel Function (2/3)

$$\text{Training}: \quad L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\Phi(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j}))$$

$$\text{Classify}: \quad y = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) + b\right)$$

$$\Downarrow$$

SVMs depend only on the evaluation of dot products

Need not to project training data if we can find the K that satisfies:

$$\Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2) \quad K : Kernel\ Function$$

Can reduce the computational overhead considerably

Kernel Function (3/3)

2nd Polynomial Function

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2 = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad \mathbf{x} \in \mathbf{R}^2 = \{x_1, x_2\}$$

$$\boldsymbol{\Phi}(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ 1 \end{pmatrix}$$

- **2** dimensional feature is projected onto **6** dimensional space
- Training with combination (co-occurance) of features
- The computational overhead dose not increase

Support Vector Machines (Summary)

- High generalization performance independent of given feature dimension
  - Maximal Margin Strategy
- Training with combinations (dependencies, co-occurence) of more than one features without increasing computational overhead
  - Use of Kernel function

$$\Downarrow$$

Effects of **smoothing** between all given features

How do we apply SVMs? (1/2)

What do we set as Positive and Negative examples?

$\Downarrow$

All candidates of two chunks which have ...

dependency relation $\rightarrow$ Positive examples

no dependency relation $\rightarrow$ Negative examples

How do we apply SVMs? (2/2)

- Dependency Probability

$$P(Dep(i) = j|\mathbf{f}_{ij}) = \tanh(\sum_{k,l} \alpha_{kl} y_{kl} K(\mathbf{f}_{kl} \cdot \mathbf{f'}_{ij}) + b)$$

$$\tanh(x) = \frac{1}{1 + \exp(-x)} \;\; (Sigmoid\ funtion)$$

- This conversion dose not give us a **true** probability, Normalizing distance $(-\infty - +\infty)$ to probability value $(0 - 1)$
- We easily apply conventional probability-based parsing techniques
- We adopted backward beam search method introduced by [Sekine 2000]

Static Features vs. Dynamic Features(1/2)

- Static Features
  - Features (Lexicon, POS, distance, postion ...) of two chunks
  - Solely defined by the pair of chunks

Static Features vs. Dynamic Features(2/2)

- Dynamic Features
  - Dependency relation themselves, added dynamicaly
  - Applying beam search to reduce the computational overhead

Experiments(1/2)

- Kyoto University Text Corpus Version 2.0
  - Training data: Articles on Jan. 1st - 7th (7958 sentences)
  - Test data: Articles on Jan. 9th (1246 sentences)
    - ∗ Same training and test data as [Uchimoto 98]
  - Kernel function: 3rd polynomial (d=3)
  - Beam width: k=5
- Evaluation method
  - Dependency accuracy
  - Sentence accuracy

Experiments(2/2)

Static Features	Left/ Right Chunks	**Head/Type** (surface-form, POS, POS-subcategory, inflection-type, inflection-form), brackets, quotation-marks, punctuation-marks, position in sentence (beginning, end)
	Between Chunks	distance(1,2-5,6-), case-particles, brackets, quotation-marks, punctuation-marks
Dynamic Features	Form of functional words or inflection that modifies the right chunk	

- The static features are basically taken from Uchimoto's 98 list
- No cut-off (frequency filter.. etc) for selecting features

Results

- Degree of Kernel Function: $d = 3$
- Beam-Width: $k = 5$

# of training sentences	Dependency Acc.	Sentence Acc.
1172	86.52%	39.31%
1917	87.21%	40.06%
3032	87.67%	42.94%
4318	88.34%	44.07%
5540	88.66%	45.20%
6756	88.77%	45.36%
7958	**89.09** %	**46.17%**

Effects of Dynamic Features

- Degree of Kernel Function: $d = 3$
- Beam-Width: $k = 5$

# of training sentences	Dynamic	without Dynamic
1172	86.52%	86.12%
1917	87.21%	86.81%
3032	87.67%	87.62%
4318	88.34%	87.33%
5540	88.66%	88.40%
6756	88.77%	88.55%
7958	**89.09**%	88.77%

Kernel Function vs. Accuracy

3,032 sentences, Beam Width: K=5

Dimension(d)	Dependency Acc.	Sentence Acc.
1	N/A	N/A
2	86.87%	40.60%
3	87.67%	**42.94%**
4	**87.72%**	42.78%

- d-th polynomial kernel $\rightarrow$
  training with all combinations of features up to d
- This results support our institution —
  The consideration of combination (dependency, co-occurance) of features is quite important for Japanese dependency analysis

Comparison with Related Work

Uchimoto 98

- Based on Maximal Entropy Model
- 87.2% (our method achieves 89.1%)
- He also pointed out the importance of considering combination (dependency, co-occurance), however these combinations are selected heuristically
  These manual selection dose not always cover all effective combinations
- The Kernel Principle allow us to build a separating hyperplane considering the any combinations of features without increasing the computational overhead

Future Works

Great amount of computational overhead is required since our proposed method uses all candidates of dependency relations

$$\Downarrow$$

Selecting only the effective portion of examples

- Introduction of (hand-crafted) constraint on non-dependency
- Integration with other simple models
- Error-driven data selection

Summary

- By applying SVMs, we can achieve a high accuracy even with a small training data (7958 sentences)
- We can show the high generalization performance and high feature selection abilities of SVMs
- The consideration of combinations (dependency, co-occurance) of features is important for Japanese dependency analysis.
  Use of Kernel functions enables feature selection more efficiently than conventional frameworks