Dependency Analysis of Japanese Spoken Language via SVM

Kiyonori Ohtake

ATR Spoken Language Translation Research Laboratories

August 6, 2003

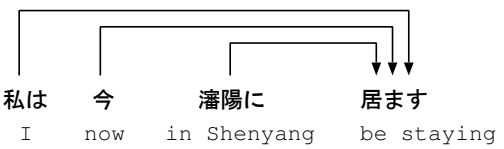
Background

- Dependency analysis for Japanese is important.
- Many works for text have been done.
- Few analyzers for spoken Japanese.
  - difficult task?
  - hard to develop analyzed spoken language corpora?

Objective

- To build a dependency analyzer for spoken language with corpus-based method.
- To prepare a consistent dependency analyzed corpus of spoken language with currently available resources.

Japanese Dependency Structure



1. From left to right.
2. Do not cross each other.
3. One segment depends on only one segment.

Dependency Analysis Approach

- Rule-based
- Corpus-based
  using ME, SVM, etc.

Why SVM?

very high generalization ability

⇓

it works well if the learning corpus is small.

Resources

- Analyzers
  - KNP: rule-based analyzer
  - CaboCha: analyzer with SVM
- Analyzed corpora
  - KUTC: newspaper articles, directly applicable to CaboCha
  - JDEP: conversations, not applicable to CaboCha

CaboCha: analyzer via SVM

Characteristics:

- Cascaded chunking model [Kudo and Matsumoto 2002]
- SVM tells whether a segment depends on the next one or not
- Available analyzed corpus is written text (KUTC)

Accuracy = 89% (by 2-fold cross validation for newspapers)

KNP: rule-based analyzer

Characteristics:

Detecting conjunctive structures, and then analyze with manually built rules.

Accuracy = 91% (for newspapers)

Corpus - KUTC

KUTC: Kyoto University Text Corpus

- newspaper articles, 40,000 sentences
- analyzed with KNP and manually corrected
- dependency segment: *bunsetu*

bunsetu: a phrasal unit that consists of one or more morphemes.

Corpus - JDEP

JDEP= Japanese Dependency structures of ATR SLDB

- travel-type conversations, 21,761 utterances
- automatically analyzed and manually corrected
- dependency segment: morpheme

Converted into KUTC format

Comparing two analyzers and two corpora

	KUTC	
	Dep. acc.	Sent. acc.
KNP	91.23%	57.17%
CaboCha	97.36%	83.40%
	JDEP	
	Dep. acc.	Sent. acc.
KNP	84.27%	69.54%
CaboCha	88.43%	76.01%

Corpus Cleaning

- Objective: to remove inconsistencies
- How: repeating closed test with CaboCha and correct manually. very SIMPLE

Standard guideline: based on KUTC annotation standard, and arranged for spoken language.

Gain by Cleaning

Evaluated by 2-fold cross validation

	Accuracy	
	KUTC	JDEP
Before	88.85%	92.78%
After	89.92%	93.03%

Discussions

- Why JDEP results were higher?
- Whether *bunsetu* is good segment for spoken language dependency analyzing?

Discussions

- Why JDEP results were higher? Sentences in JDEP are shorter.
- Whether *bunsetu* is good segment for spoken language dependency analyzing?

Discussions

- Why JDEP results were higher? Sentences in JDEP are shorter.
- Whether *bunsetu* is good segment for spoken language dependency analyzing?
  $\Rightarrow$ No. Need to investigate further.

Open problems

- Much learning time.
- To prepare properly segmented corpus.
- How to analyze real spoken language.

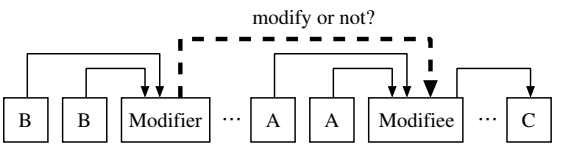
Conclusion

- Introduced available resources, and built an analyzer via SVM.
- Carried out the corpus cleaning.
- Cleaning raised the acc. (1.07% for KUTC, 0.25% for JDEP).
- The analyzer resulted 93% acc. for spoken language.

Thank you!

Appendix - Features for SVM

Dynamic features:



Static features: head words, parts-of-speech, functional words and inflection forms, and so on.

Appendix - cascaded chunking

私は	今	中国の	瀋陽に	居ます
O	O	O	O	O

Step 1. Initialization

O tag means the dependency relation is undecided.

Appendix - cascaded chunking

私は	今	中国の	瀋陽に	居ます
O	O	D	D	O

Step 2. Each **O** tag depends on immediate right hand side?
⇒ replace with **D**

Appendix - cascaded chunking

私は	今	中国の	瀋陽に	居ます
O	O	D	D	O
		delete		

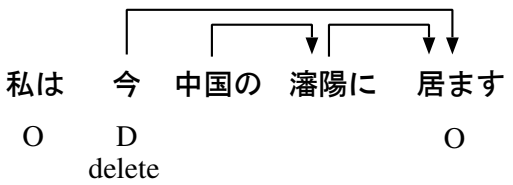
Step 3. Delete segments with **D** tag that immediately follow **O** tag.

Appendix - cascaded chunking

私は	今	中国の	瀋陽に	居ます
O	O		D	O
			delete	

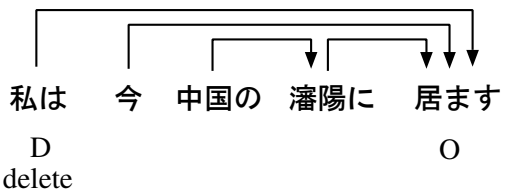
Repeating Step 2 and 3.

Appendix - cascaded chunking



Repeating Step 2 and 3.

Appendix - cascaded chunking



Repeating Step 2 and 3.
And finished.