Japanese Language Processing

Symbols and Formal Models

Petr Horáček

Department of Information Systems
Faculty of Information Technology
Brno University of Technology

December 31, 2009

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database

2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database

2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

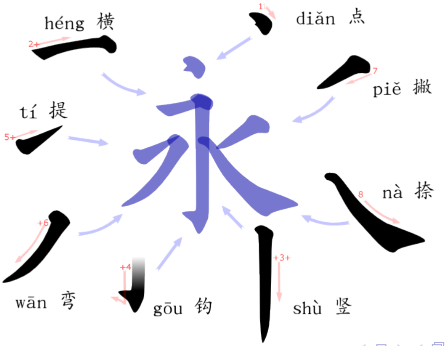
Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database

2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Writing *Kanji* Characters

- *Kanji* (漢字) are Chinese characters used in Japanese writing system.
- Each character is composed of several strokes.
- There are 8 basic strokes, shown in the chinese character *yong* (eternity) below.
- Several basic strokes can combine into one compound stroke.
- There is a given stroke order for writing each character.



Kanji Character Recognition

- Character recognition can be useful in many practical applications, such as:
  - Learning applications - Test if the user has learned to write the *kanji*.
  - Dictionaries - Even if the user is unfamiliar with Japanese, they may be able to redraw the character they want to look up.

Problem

- If we want to classify the character's image as a whole, there is too many classes (characters).

Solution

- If we have a vector representation of the character, we can recognize it by classifying the strokes and analyzing their relations (mainly positions) inside the character.
  - There is only a relatively small number of strokes.
  - The strokes can be easily distinguished.

Kanji Character Recognition

- Character recognition can be useful in many practical applications, such as:
  - Learning applications - Test if the user has learned to write the *kanji*.
  - Dictionaries - Even if the user is unfamiliar with Japanese, they may be able to redraw the character they want to look up.

Problem

- If we want to classify the character's image as a whole, there is too many classes (characters).

Solution

- If we have a vector representation of the character, we can recognize it by classifying the strokes and analyzing their relations (mainly positions) inside the character.
  - There is only a relatively small number of strokes.
  - The strokes can be easily distinguished.

Kanji Character Recognition

- Character recognition can be useful in many practical applications, such as:
  - Learning applications - Test if the user has learned to write the *kanji*.
  - Dictionaries - Even if the user is unfamiliar with Japanese, they may be able to redraw the character they want to look up.

Problem

- If we want to classify the character's image as a whole, there is too many classes (characters).

Solution

- If we have a vector representation of the character, we can recognize it by classifying the strokes and analyzing their relations (mainly positions) inside the character.
  - There is only a relatively small number of strokes.
  - The strokes can be easily distinguished.

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database
2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Kanji Character Semantics

- A single character typically has several readings, and can have more than one meaning.
- Readings are divided into 2 groups:
  1. *On'yomi* (音読み) - Sino-Japanese reading, a Japanese approximation of a Chinese pronunciation. Used mostly for compound words (consisting of more than one *kanji*).
  2. *Kun'yomi* (訓読み) - Japanese reading, based on a native Japanese word with a similar meaning. Used mostly for stand-alone *kanji* characters.

Example

Character	*On'yomi*	*Kun'yomi*	Meanings
行	KOU, GYOU コウ，ギョウ	iku, yuku い(く)，ゆ(く)	to go

Kanji Character Semantics

- A single character typically has several readings, and can have more than one meaning.
- Readings are divided into 2 groups:
  1. *On'yomi* (音読み) - Sino-Japanese reading, a Japanese approximation of a Chinese pronunciation. Used mostly for compound words (consisting of more than one *kanji*).
  2. *Kun'yomi* (訓読み) - Japanese reading, based on a native Japanese word with a similar meaning. Used mostly for stand-alone *kanji* characters.

Example

Character	*On'yomi*	*Kun'yomi*	Meanings
行	KOU, GYOU コウ，ギョウ	iku, yuku い(く)，ゆ(く)	to go

Kanji Character Dependencies

- A sequence of *kanji* characters (usually 2, but it can be more) can form a compound word with a new meaning.
- This also determines the readings of the *kanji* in the compound word.

Example

Character	Readings	Meanings	Compounds
日	NICHI, JITSU ニチ，ジツ hi, ka ひ，か	the sun day	ある日 *aruhi* - one day 毎日 *mainichi* - everyday 今日 *kyou* (irregular) - today 日曜日 *nichiyoubi* - Sunday
学	GAKU, KAKU ガク，カク manabu まな(ぶ)	learning science to learn	科学 *kagaku* - science 学校 *gakkou* - school 数学 *suugaku* - math

Kanji Character Dependencies

- A sequence of *kanji* characters (usually 2, but it can be more) can form a compound word with a new meaning.
- This also determines the readings of the *kanji* in the compound word.

Example

Character	Readings	Meanings	Compounds
日	NICHI, JITSU ニチ，ジツ hi, ka ひ，か	the sun day	ある日 *aruhi* - one day 毎日 *mainichi* - everyday 今日 *kyou* (irregular) - today 日曜日 *nichiyoubi* - Sunday
学	GAKU, KAKU ガク，カク manabu まな(ぶ)	learning science to learn	科学 *kagaku* - science 学校 *gakkou* - school 数学 *suugaku* - math

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database

2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Building a *Kanji* Database

The minimal information that we need to maintain for each character contains:

1. Readings
   - We should distinguish between *kon'yomi* and *on'yomi*
2. Meanings
3. Relations and dependencies to other characters
   - Compounds
   - Radicals
4. Graphical representation
   - Preferably vector-based

Character relations are very important, since they affect the characters' readings and meanings.

- Create a relational model?
- Implement as a relational database (SQL. . .)?

Building a *Kanji* Database

The minimal information that we need to maintain for each character contains:

1. Readings
   - We should distinguish between *kon'yomi* and *on'yomi*
2. Meanings
3. Relations and dependencies to other characters
   - Compounds
   - Radicals
4. Graphical representation
   - Preferably vector-based

Character relations are very important, since they affect the characters' readings and meanings.

- Create a relational model?
- Implement as a relational database (SQL. . .)?

Building a *Kanji* Database

The minimal information that we need to maintain for each character contains:

1. Readings
   - We should distinguish between *kon'yomi* and *on'yomi*
2. Meanings
3. Relations and dependencies to other characters
   - Compounds
   - Radicals
4. Graphical representation
   - Preferably vector-based

Character relations are very important, since they affect the characters' readings and meanings.

- Create a relational model?
- Implement as a relational database (SQL. . .)?

Building a *Kanji* Database

The minimal information that we need to maintain for each character contains:

1. Readings
   - We should distinguish between *kon'yomi* and *on'yomi*
2. Meanings
3. Relations and dependencies to other characters
   - Compounds
   - Radicals
4. Graphical representation
   - Preferably vector-based

Character relations are very important, since they affect the characters' readings and meanings.

- Create a relational model?
- Implement as a relational database (SQL...)?

Building a *Kanji* Database

The minimal information that we need to maintain for each character contains:

1. Readings
    - We should distinguish between *kon'yomi* and *on'yomi*
2. Meanings
3. Relations and dependencies to other characters
    - Compounds
    - Radicals
4. Graphical representation
    - Preferably vector-based

Character relations are very important, since they affect the characters' readings and meanings.

- Create a relational model?
- Implement as a relational database (SQL. . .)?

Outline

Natural Language Processing and Context-Free Grammars

- Formal models practically used in natural language processing (NLP) are often based on context-free grammars.

Definition

A context-free grammar (CFG) is a quadruple $G = (N, T, P, S)$, where

- N is a finite set of *nonterminal* symbols
- T is a finite set of *terminal* symbols, $N \cap T = \emptyset$
- P is a finite relation from N to $(N \cup T)^*$, usually represented as a finite set of *rules (productions)* of the form

  $$A \rightarrow x,$$

  where $A \in N$ and $x \in (N \cup T)^*$
- $S \in N$ is the *start symbol*

Context-Free Grammar

Derivation

1. Let $G = (N, T, P, S)$ be a CFG. Let $u, v \in (N \cup T)^*$ and $p = A \rightarrow x \in P$. Then, uAv *directly derives* uxv according to p in G, written as $uAv \Rightarrow_G uxv \; [p]$ or simply $uAv \Rightarrow uxv$.
2. The relation $\Rightarrow^*$ *(derives)* is the reflexive transitive closure of $\Rightarrow$.

Generated language

Let $G = (N, T, P, S)$ be a CFG. The *language generated by* G, denoted as $L(G)$, is defined as

$$L(G) = \{w : w \in T^*, S \Rightarrow^* w\}$$

Beyond CFG

Problem

- CFG by itself has insufficient generative power to describe a natural language (natural languages are generally not context-free).
- Context-sensitive (Type-1) and general (Type-0) grammars are not suitable for practical implementation.

Solution

- Increase the generative power without significantly increasing the implemenatation cost. (Find "something between" CF and CS?)
- New models are often based on CFG (or context-free rules).
  - Common in NLP: CCG (using combinatory logic), PCFG/SCFG (probabilistic CFG, using statistical approach), LTAG (rewriting tree nodes instead of symbols)...
  - Other: matrix grammar, random context grammar, programmed grammar, scattered context grammar...

Beyond CFG

Problem

- CFG by itself has insufficient generative power to describe a natural language (natural languages are generally not context-free).
- Context-sensitive (Type-1) and general (Type-0) grammars are not suitable for practical implementation.

Solution

- Increase the generative power without significantly increasing the implemenatation cost. (Find "something between" CF and CS?)
- New models are often based on CFG (or context-free rules).
  - Common in NLP: CCG (using combinatory logic), PCFG/SCFG (probabilistic CFG, using statistical approach), LTAG (rewriting tree nodes instead of symbols). . .
  - Other: matrix grammar, random context grammar, programmed grammar, scattered context grammar. . .

Matrix Grammar

Definition

A matrix grammar is a pair $H = (G, M)$, where

- $G = (N, T, P, S)$ is a context-free grammar
- M is a finite language over P $(M \subseteq P^*)$

Notation

- Let $N = A_1, \ldots, A_m$ for some $m \geq 1$
- For some $m_i = p_{i_1} \ldots p_{i_j} \ldots p_{i_{k_i}} \in M$,

$$p_{i_j} : A_{i_j} \rightarrow x_{i_j}$$

Matrix Grammar

Derivation step

For $x, y \in (N \cup T)^*$, $m \in M$,

$$x \Rightarrow y[m]$$

in H if there are $x_0, \ldots, x_n$ such that $x = x_0, x_n = y$, and

1. $x_0 \Rightarrow x_1[p_1] \Rightarrow x_2[p_2] \Rightarrow \cdots \Rightarrow x_n[p_n]$ in G, and
2. $m = p_1 \ldots p_n$

Generated language

$$L(H) = \{x \in T^* : S \Rightarrow^* x\}$$

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database
2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	→	SP OP VP	5:	NP	→	N
2:	SP	→	NP *wa*	6:	N	→	*kore* | *daigaku*
3:	OP	→	NP	7:	V	→	*desu*
4:	VP	→	V				

- One of the possible derivations is:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V ⇒ *kore wa* N V ⇒ *kore wa daigaku* V ⇒ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V ⇒ *kore wa* N V ⇒ *kore wa daigaku* V ⇒ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V ⇒ *kore wa* N V ⇒ *kore wa daigaku* V ⇒ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	→	SP OP VP	5:	NP	→	N
2:	SP	→	NP *wa*	6:	N	→	*kore* | *daigaku*
3:	OP	→	NP	7:	V	→	*desu*
4:	VP	→	V				

- One of the possible derivations is:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V ⇒ *kore wa* N V ⇒ *kore wa daigaku* V ⇒ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Describing Japanese Language

- Japanese sentences are in the SOV (subject-object-verb) form.
  - English, for example, uses SVO.
- Consider a CFG G with the following rules (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	5:	NP	$\rightarrow$	N
2:	SP	$\rightarrow$	NP *wa*	6:	N	$\rightarrow$	*kore* | *daigaku*
3:	OP	$\rightarrow$	NP	7:	V	$\rightarrow$	*desu*
4:	VP	$\rightarrow$	V				

- One of the possible derivations is:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* NP VP $\Rightarrow$ N *wa* N VP $\Rightarrow$ N *wa* N V $\Rightarrow$ *kore wa* N V $\Rightarrow$ *kore wa daigaku* V $\Rightarrow$ *kore wa daigaku desu* ("this is an university")
  which is a well-formed Japanese sentence. Likewise, all other successful derivations in G produce a well-formed sentence.

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to G:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to G:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \quad | \quad 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to *G*:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to *G*:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \quad | \quad 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to G:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to G:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \quad | \quad 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to *G*:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to *G*:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \;\Big|\; 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to *G*:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V}\ \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to *G*:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \quad \Big| \quad 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to *G*:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to *G*:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \;\Big|\; 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to *G*:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V } \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to *G*:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \;\Big|\; 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to G:

$$8: \quad VP \quad \rightarrow \quad V \ \mathit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to G:

$$9: \quad OP \quad \rightarrow \quad INT \ \Big| \ 10: \quad INT \quad \rightarrow \quad \mathit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions

- To change a statement into a question, we can simply append *ka* at the end of the sentence.
- We can add the following rule to G:

$$8: \quad \text{VP} \quad \rightarrow \quad \text{V}\ \textit{ka}$$

- Derivation example:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* NP VP ⇒ N *wa* N VP ⇒ N *wa* N V *ka* ⇒ *kore wa* N V *ka* ⇒ *kore wa daigaku* V *ka* ⇒ *kore wa daigaku desu ka* ("is this an university?")
- Adding the following 2 rules to G:

$$9: \quad \text{OP} \quad \rightarrow \quad \text{INT} \;\Big|\; 10: \quad \text{INT} \quad \rightarrow \quad \textit{nan}$$

  allows us to generate: S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka* ("what is this?")

Forming Questions - Problem

- However, after adding rules 9 and 10, the following derivation also becomes possible:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V ⇒ *kore wa* INT V ⇒ *kore wa nan* V ⇒ *kore wa nan desu* ("this is what")
- But "*kore wa nan desu*" is not a well-formed sentence. We need to modify the grammar so that it does not allow this derivation.
- With CFG only, this is complicated and we would need to add more rules and nonterminals.
- We can construct a matrix grammar $H = (G, M)$, where G is the discussed CFG.
  - We do not need any additional rules or nonterminals.

Forming Questions - Problem

- However, after adding rules 9 and 10, the following derivation also becomes possible:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V ⇒ *kore wa* INT V ⇒ *kore wa nan* V ⇒ *kore wa nan desu* ("this is what")
- But "*kore wa nan desu*" is not a well-formed sentence. We need to modify the grammar so that it does not allow this derivation.
- With CFG only, this is complicated and we would need to add more rules and nonterminals.
- We can construct a matrix grammar $H = (G, M)$, where G is the discussed CFG.
  - We do not need any additional rules or nonterminals.

Forming Questions - Problem

- However, after adding rules 9 and 10, the following derivation also becomes possible:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V ⇒ *kore wa* INT V ⇒ *kore wa nan* V ⇒ *kore wa nan desu* ("this is what")
- But "*kore wa nan desu*" is not a well-formed sentence. We need to modify the grammar so that it does not allow this derivation.
- With CFG only, this is complicated and we would need to add more rules and nonterminals.
- We can construct a matrix grammar $H = (G, M)$, where G is the discussed CFG.
  - We do not need any additional rules or nonterminals.

Forming Questions - Problem

- However, after adding rules 9 and 10, the following derivation also becomes possible:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT VP ⇒ N *wa* INT V ⇒ *kore wa* INT V ⇒ *kore wa nan* V ⇒ *kore wa nan desu* ("this is what")
- But "*kore wa nan desu*" is not a well-formed sentence. We need to modify the grammar so that it does not allow this derivation.
- With CFG only, this is complicated and we would need to add more rules and nonterminals.
- We can construct a matrix grammar $H = (G, M)$, where G is the discussed CFG.
  - We do not need any additional rules or nonterminals.

Describing Japanese Language Using Matrix Grammar

- Consider a matrix grammar $H = (G, M)$ with the following rules in G (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	6:	N	$\rightarrow$	*kore* | *daigaku*
2:	SP	$\rightarrow$	NP *wa*	7:	V	$\rightarrow$	*desu*
3:	OP	$\rightarrow$	NP	8:	VP	$\rightarrow$	V *ka*
4:	VP	$\rightarrow$	V	9:	OP	$\rightarrow$	INT
5:	NP	$\rightarrow$	N	10:	INT	$\rightarrow$	*nan*

  and $M = \{(1), (2), (3), (4), (5), (6), (7), (8), (9, 8), (10)\}$

- The derivation in question in H becomes:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* INT V *ka* $\Rightarrow$ *kore wa* INT V *ka* $\Rightarrow$ *kore wa nan* V *ka* $\Rightarrow$ *kore wa nan desu ka*
  ("what is this?")
  and it is no longer possible to generate an invalid sentence.

Describing Japanese Language Using Matrix Grammar

- Consider a matrix grammar $H = (G, M)$ with the following rules in G (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	6:	N	$\rightarrow$	*kore* | *daigaku*
2:	SP	$\rightarrow$	NP *wa*	7:	V	$\rightarrow$	*desu*
3:	OP	$\rightarrow$	NP	8:	VP	$\rightarrow$	V *ka*
4:	VP	$\rightarrow$	V	9:	OP	$\rightarrow$	INT
5:	NP	$\rightarrow$	N	10:	INT	$\rightarrow$	*nan*

  and $M = \{(1), (2), (3), (4), (5), (6), (7), (8), (9, 8), (10)\}$

- The derivation in question in H becomes:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka*
  ("what is this?")
  and it is no longer possible to generate an invalid sentence.

Describing Japanese Language Using Matrix Grammar

- Consider a matrix grammar $H = (G, M)$ with the following rules in G (nonterminals are in capital letters):

1:	S	$\rightarrow$	SP OP VP	6:	N	$\rightarrow$	*kore* | *daigaku*
2:	SP	$\rightarrow$	NP *wa*	7:	V	$\rightarrow$	*desu*
3:	OP	$\rightarrow$	NP	8:	VP	$\rightarrow$	V *ka*
4:	VP	$\rightarrow$	V	9:	OP	$\rightarrow$	INT
5:	NP	$\rightarrow$	N	10:	INT	$\rightarrow$	*nan*

  and $M = \{(1), (2), (3), (4), (5), (6), (7), (8), (9, 8), (10)\}$

- The derivation in question in H becomes:
  S $\Rightarrow$ SP OP VP $\Rightarrow$ NP *wa* OP VP $\Rightarrow$ N *wa* OP VP $\Rightarrow$ N *wa* INT V *ka* $\Rightarrow$ *kore wa* INT V *ka* $\Rightarrow$ *kore wa nan* V *ka* $\Rightarrow$ *kore wa nan desu ka* ("what is this?")
  and it is no longer possible to generate an invalid sentence.

Describing Japanese Language Using Matrix Grammar

- Consider a matrix grammar $H = (G, M)$ with the following rules in G (nonterminals are in capital letters):

1:	S	→	SP OP VP	6:	N	→	*kore* | *daigaku*
2:	SP	→	NP *wa*	7:	V	→	*desu*
3:	OP	→	NP	8:	VP	→	V *ka*
4:	VP	→	V	9:	OP	→	INT
5:	NP	→	N	10:	INT	→	*nan*

  and $M = \{(1), (2), (3), (4), (5), (6), (7), (8), (9, 8), (10)\}$

- The derivation in question in H becomes:
  S ⇒ SP OP VP ⇒ NP *wa* OP VP ⇒ N *wa* OP VP ⇒ N *wa* INT V *ka* ⇒ *kore wa* INT V *ka* ⇒ *kore wa nan* V *ka* ⇒ *kore wa nan desu ka*
  ("what is this?")
  and it is no longer possible to generate an invalid sentence.

Outline

1. Processing *kanji* characters
   - Writing and character recognition
   - Character semantics and dependencies
   - Building a *kanji* database
2. Processing Japanese text
   - Natural language processing and formal models
   - Describing Japanese language
   - Specific problems

Specific Problems

- Besides *kanji*, the Japanese writing system also uses 2 syllabaries:
  1. *Hiragana* (ひらがな) - used for particles, suffixes. Also can be used for words that have no *kanji*, or if the *kanji* is not known to the writer. It is also possible to write Japanese text entirely in *hiragana* (without using *kanji*; this is common in children's books, japanese textbooks for beginners etc.)
  2. *Katakana* (カタカナ) - used mainly for transcribing words from foreign languages (including names).
- Modern Japanese writing sometimes also includes Latin alphabet (mostly abbreviations) and roman numbers.
- The transcription of Japanese text using Latin alphabet is called *romaji* (ローマ字). There are several different standards.
- In Japanese sentences, it is not customary to separate words by spaces.

Specific Problems

Example

- A typical Japanese sentence might look like this:
  日本とチェコスロバキア間の外交関係は
  1919年に樹立されました。
  ("The diplomatic relations between Japan and Czechoslovakia started in 1919.")
- However, it is also possible to write it in *hiragana* only:
  にほんとちぇこすろばきあかんのがいこうかんけいは
  1919ねんにじゅりつされました。
- If we transcribe it into *romaji*, it may look like this:
  Nihon to Chekosurobakia kan no gaikou kankei wa
  1919 nen ni juritsu saremashita.

Specific Problems

Example

- A typical Japanese sentence might look like this:
  日本とチェコスロバキア間の外交関係は
  1919年に樹立されました。
  ("The diplomatic relations between Japan and Czechoslovakia started in 1919.")
- However, it is also possible to write it in *hiragana* only:
  にほんとちぇこすろばきあかんのがいこうかんけいは
  1919ねんにじゅりつされました。
- If we transcribe it into *romaji*, it may look like this:
  Nihon to Chekosurobakia kan no gaikou kankei wa
  1919 nen ni juritsu saremashita.

Specific Problems

Example

- A typical Japanese sentence might look like this:
  日本とチェコスロバキア間の外交関係は
  1919年に樹立されました。
  ("The diplomatic relations between Japan and Czechoslovakia started in 1919.")
- However, it is also possible to write it in *hiragana* only:
  にほんとちぇこすろばきあかんのがいこうかんけいは
  1919ねんにじゅりつされました。
- If we transcribe it into *romaji*, it may look like this:
  Nihon to Chekosurobakia kan no gaikou kankei wa
  1919 nen ni juritsu saremashita.

Specific Problems

Problems

1. We have to separate the sentence into words (syntactic units).
2. Either of the previous examples is a valid way to write the same sentence (and there are more ways. . .). If we do not want to impose limits on the input, we should be able to parse all of these notations.
   - We can either parse the input directly, or first convert it to a notation that we are able to parse.

Solution

- Any suggestions?

Specific Problems

Problems

1. We have to separate the sentence into words (syntactic units).
2. Either of the previous examples is a valid way to write the same sentence (and there are more ways. . .). If we do not want to impose limits on the input, we should be able to parse all of these notations.
   - We can either parse the input directly, or first convert it to a notation that we are able to parse.

Solution

- Any suggestions?

Specific Problems

Problems

1. We have to separate the sentence into words (syntactic units).
2. Either of the previous examples is a valid way to write the same sentence (and there are more ways. . .). If we do not want to impose limits on the input, we should be able to parse all of these notations.
   - We can either parse the input directly, or first convert it to a notation that we are able to parse.

Solution

- Any suggestions?

References

- A. Meduna, J. Techet: Scattered Context Grammars and their Applications, WIT Press, 2009
- S. Ábrahám: Some questions of language theory, International Conference on Computational Linguistic, 1965
- J. Dassow, Gh. Păun: Regulated Rewriting in Formal Language Theory, Akademie-Verlag, Berlin, 1989.
- E. Banno, Y. Ohno, Y. Sakane, C. Shinagawa: Genki 1: An Integrated Course in Elementary Japanese, The Japan Times, 1999
- C. Kano, H. Takenaka, E. Ishii, Y. Shimizu: Basic Kanji Book, Bonjinsha, 1990
- C. D. Manning, H. Schütze: Foundations of Statistical Natural Language Processing, MIT Press, 1999