

Japanese Kanji Suggestion Tool

Sujata Dongre

CS298

San Jose State University

Outline

- Introduction
- Prior work in Japanese word segmentation
- Hidden Markov Model for text parsing
- Design and implementation
- Experiments and results
- Conclusion

Introduction

- Motivation
 - “No search results found” message on typing wrong kanjis
 - Meaningless translations of wrong Japanese word
- Goal
 - Provide simple suggestions to Japanese language beginners

Prior work in Japanese word segmentation

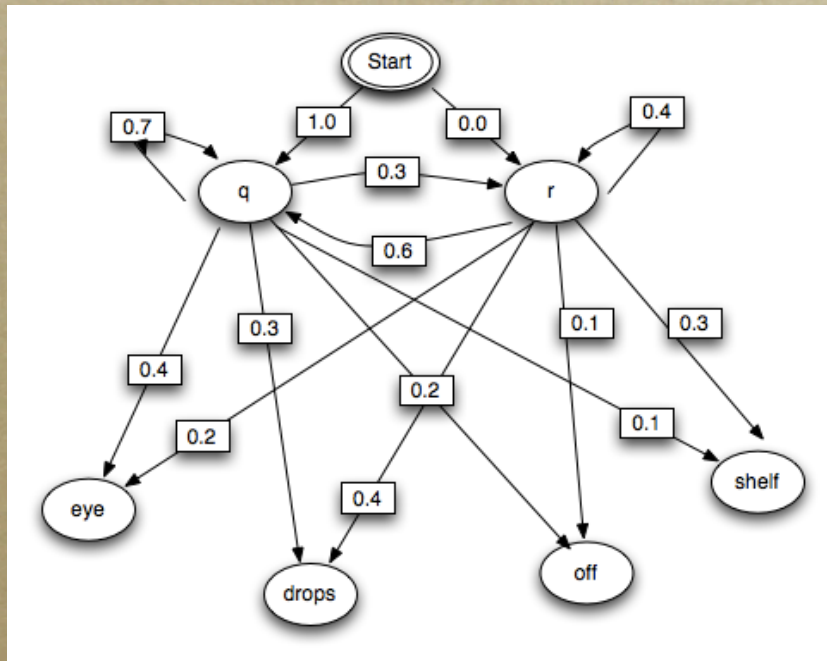
- JUMAN morphological analyzer
 - Rule-based morphological analyzer
 - Cost to lexical entry and cost to pairs of adjacent parts-of-speech
 - labor-intensive and vulnerable to unknown word problem
- TANGO algorithm
 - Based on 4-gram approach
 - Series of questions to get a word boundary
 - More robust and portable to other domains and applications

Prior work in Japanese word segmentation (cont..)

- Existing search engines
 - Google
 - Yahoo!
 - Bing

Hidden Markov Model for text parsing

- What is the Hidden Markov Model?
 - It is a variant of a finite state machine having a set of hidden states



$N =$ the number of states

$M =$ the number of observation symbols

$Q = \{q_i\}, i = 1, \dots, N$

$A =$ the state transition probabilities

$B =$ the observation probability matrix

$\Pi =$ the initial state distribution

$O = \{o_k\}, k = 1, \dots, M$

Hidden Markov Model for text parsing (cont..)

- Working of the Hidden Markov Model
 - Three problems related to the Hidden Markov Model
 1. Given the model λ and a sequence of observations, find out the sequence of hidden states that leads to the given set of observations - Viterbi algorithm
 2. Given the model λ and a sequence of observations, find out the probability of a sequence of observations - Forward or Backward algorithm
 3. Given an observation sequence O and the dimensions N and M , find the model $\lambda = (A, B, \pi)$, that maximizes the probability of O - Baum-Welch algorithm or HMM training

Design and implementation

- Japanese language processing
 - Hiragana, katakana and kanji
 - Japanese characters encoding
- Hidden Markov Model program details
 - Number of iterations
 - Number of observations
 - Number of states

Design and implementation (cont..)

- Japanese corpus - Tanaka
 - Corpus file format

A: &という記号は、 a n d を指す。 [TAB]The sign '&' stands for 'and'.#ID=1

B: と言う { という } ~ 記号 ~ は を 指す [03] ~
 - Modifications in the corpus file
- The software
 - JDK1.6, Tomcat 5.5, Eclipse IDE

Design and implementation (cont..)

- The Nutch web crawler (GUI)
 - Open source web crawler
 - Domain name to crawl japanese websites, google.co.jp
 - Command to crawl:

bin/nutch crawl urls -dir crawljp -depth 3 -topN 10

-depth: Indicates the link depth from the root page that should be crawled

-topN: Determines the maximum number of pages that will be retrieved at each level up to the depth

- Agent name in nutch-domain.xml as google

Design and implementation (cont..)

- Searcher.dir property tag in nutch-site.xml as path to crawljp directory
- Instant search functionality: Find-as-you-type

Experiments and results

- Hidden Markov Model - English text
 - Understanding how the Hidden Markov Model converges
 - Distinguish between consonants and vowels, letters a, e, i, o, u have the highest probabilities and appears in the first state
 - The observation 'space' has the highest probability among all 27 observations

Experiments and results (cont..)

- Hidden Markov Model - Japanese text
 - Frequently used characters (あ、い、う、お、で、の):
higher probabilities but no clear distinction for word boundaries
 - HMM final probability matrices are serializable and stored in a file
 - Viterbi program reads serialized object from a file and appends hiragana characters at the end of the user input string
 - Verify the string returned from Viterbi program exists in Tanaka Corpus

Experiments and results (cont..)

- N-gram experiments using Tanaka Corpus
 1. Experiment 1:
 - ▶ Aim: To find suggestions for a possible next character
 - ▶ Results: List of the first three most common words that begin with the user entered string
 - ▶ Description:
 - Binary tree node consists of <key(word of length 3), value (number of occurrences)> pair
 - Any special character is stored as 'EOW' (End Of Word)

Experiments and results (cont..)

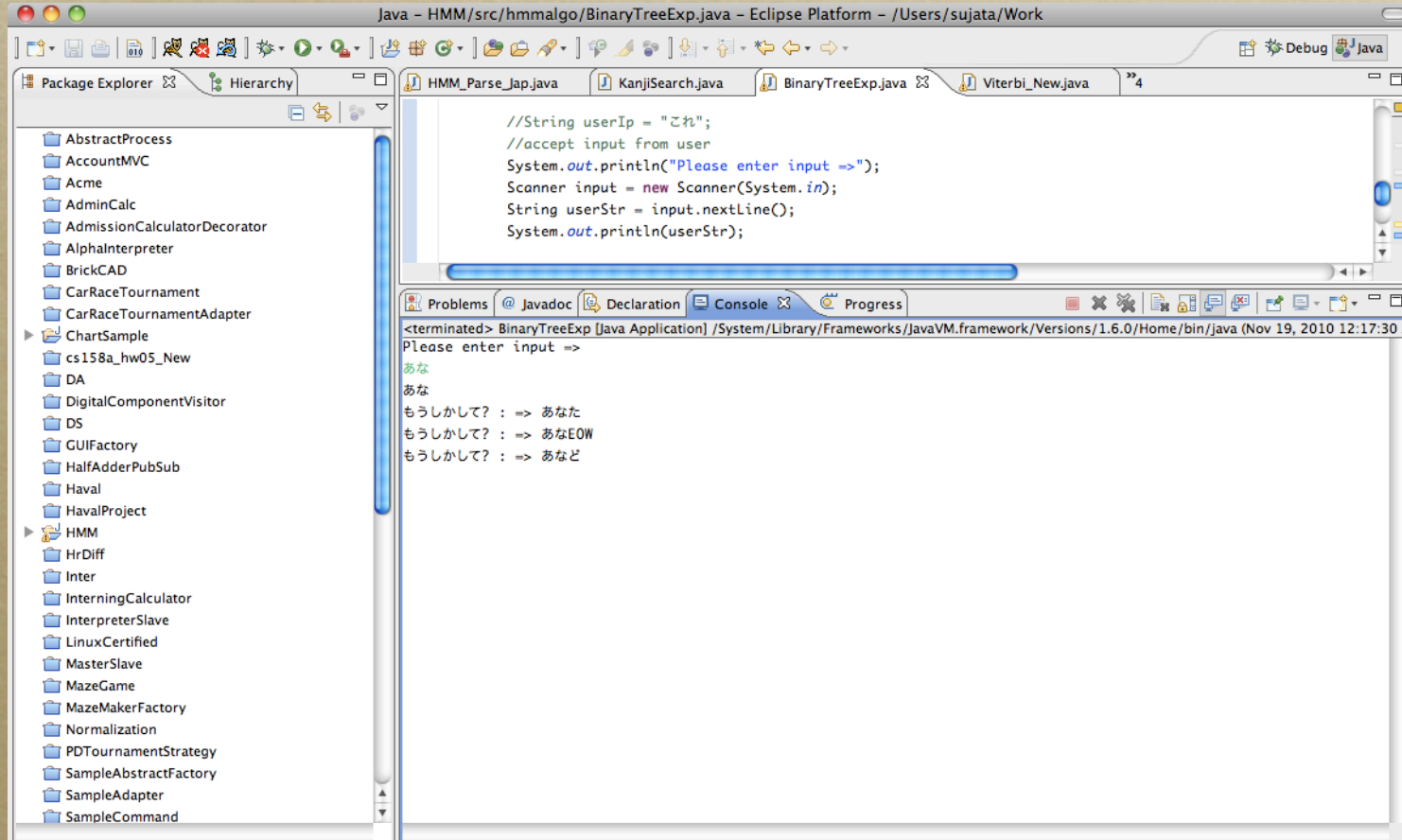
1. Experiment 1:

▶ Description:

- When user enters the input, look for the words starting with the user input and having the highest number of occurrences

Experiments and results (cont..)

1. Experiment 1:



```
Java - HMM/src/hmmalgo/BinaryTreeExp.java - Eclipse Platform - /Users/sujata/Work  
Package Explorer | Hierarchy | HMM_Parse_Jap.java | KanjiSearch.java | BinaryTreeExp.java | Viterbi_New.java | 4  
AbstractProcess  
AccountMVC  
Acme  
AdminCalc  
AdmissionCalculatorDecorator  
AlphaInterpreter  
BrickCAD  
CarRaceTournament  
CarRaceTournamentAdapter  
ChartSample  
cs158a_hw05_New  
DA  
DigitalComponentVisitor  
DS  
GUIFactory  
HalfAdderPubSub  
Haval  
HavalProject  
HMM  
HrDiff  
Inter  
InteriningCalculator  
InterpreterSlave  
LinuxCertified  
MasterSlave  
MazeGame  
MazeMakerFactory  
Normalization  
PDTournamentStrategy  
SampleAbstractFactory  
SampleAdapter  
SampleCommand  
//String userInput = "これ";  
//accept input from user  
System.out.println("Please enter input =>");  
Scanner input = new Scanner(System.in);  
String userStr = input.nextLine();  
System.out.println(userStr);  
Problems | Javadoc | Declaration | Console | Progress  
<terminated> BinaryTreeExp [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (Nov 19, 2010 12:17:30 A  
Please enter input =>  
あな  
あな  
もうしかして? : => あなた  
もうしかして? : => あなEOW  
もうしかして? : => あなど
```


Experiments and results (cont..)

2. Experiment 2:

- ▶ Aim: To find out word boundaries
- ▶ Results: Single word that begin with the user entered string
- ▶ Description:
 - Iterate through Tanaka Corpus reading string of length three
 - String ending with the special character: subtract 1 else add 1
 - Find out words having positive number of occurrences indicating end of word

Experiments and results (cont..)

2. Experiment 2:

```
Java - HMM/src/hmmalgo/HMM_Parse_Jap.java - Eclipse Platform - /Users/sujata/Work  
Package Explorer | Hierarchy | HMM_Parse_Jap.java | KanjiSearch.java | BinaryTreeExp.java | Viterbi_New.java | 4  
AbstractProcess  
AccountMVC  
Acme  
AdminCalc  
AdmissionCalculatorDecorator  
AlphaInterpreter  
BrickCAD  
CarRaceTournament  
CarRaceTournamentAdapter  
ChartSample  
cs158a_hw05_New  
DA  
DigitalComponentVisitor  
DS  
GUIFactory  
HalfAdderPubSub  
Haval  
HavalProject  
HMM  
HrDiff  
Inter  
InterningCalculator  
InterpreterSlave  
LinuxCertified  
MasterSlave  
MazeGame  
MazeMakerFactory  
Normalization  
PDTournamentStrategy  
SampleAbstractFactory  
SampleAdapter  
SampleCommand  
//call dictionary window program  
HashMap<String, Integer> hmDicto = DictionaryWindow_New.createDicto();  
//accept input from user  
System.out.println("Please enter input of length 2 =>");  
Scanner inputWindow = new Scanner(System.in);  
String userStrWindow = inputWindow.nextLine();  
System.out.println(userStrWindow);  
  
//search if this user input exists in hmDicto with positive count  
Problems | @ Javadoc | Declaration | Console | Progress  
<terminated> HMM_Parse_Jap [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (Nov 19, 2010 12:14:27)  
logProb => -488053.37265388446  
oldLogProb => -488053.44983108435  
Please enter input of length 2 =>  
あなた  
あなた  
cnt => -9580  
*****1. もうしかして? :あなた
```

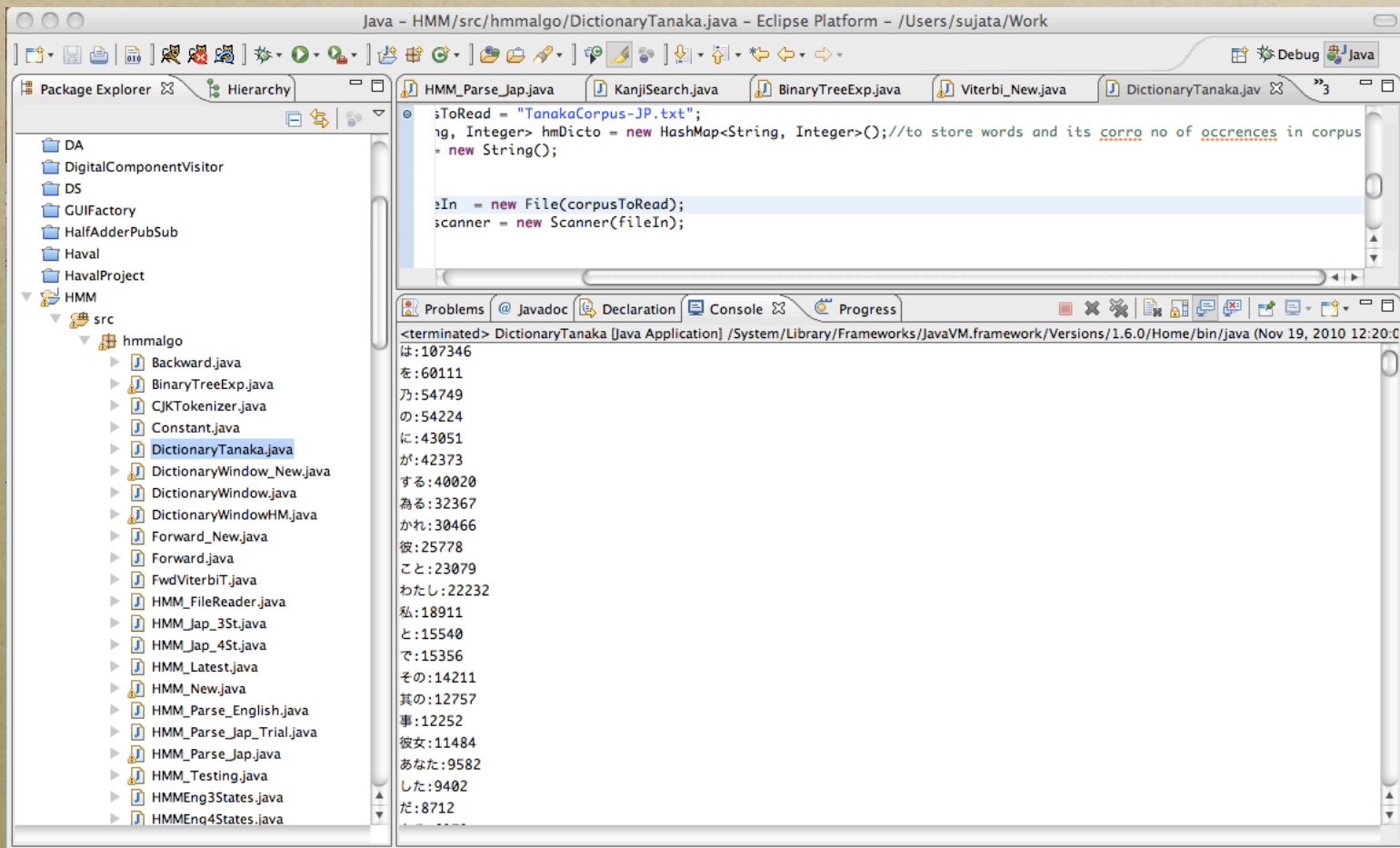

Experiments and results (cont..)

3. Experiment 3:

- ▶ Aim: To find out all Japanese words in the corpus file
- ▶ Results: List of Japanese words
- ▶ Description:
 - Creates Japanese word dictionary
 - Can be used in information security

Experiments and results (cont..)

3. Experiment 3:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, including a 'DictionaryTanaka.java' file. The main editor shows the code for this file, which reads a corpus file and counts the occurrences of various words. The Console window at the bottom displays the output of the application, listing words and their counts.

```
DictionaryTanaka.java  
:toRead = "TanakaCorpus-JP.txt";  
ng, Integer> hmDicto = new HashMap<String, Integer>(); //to store words and its corro no of occrances in corpus  
= new String();  
  
>In = new File(corpusToRead);  
scanner = new Scanner(fileIn);
```

Console Output:

```
<terminated> DictionaryTanaka [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (Nov 19, 2010 12:20:0  
は:107346  
を:60111  
乃:54749  
の:54224  
に:43051  
が:42373  
する:40020  
為る:32367  
かれ:30466  
彼:25778  
こと:23079  
わたし:22232  
私:18911  
と:15540  
で:15356  
その:14211  
其の:12757  
事:12252  
彼女:11484  
あなた:9582  
した:9402  
だ:8712
```

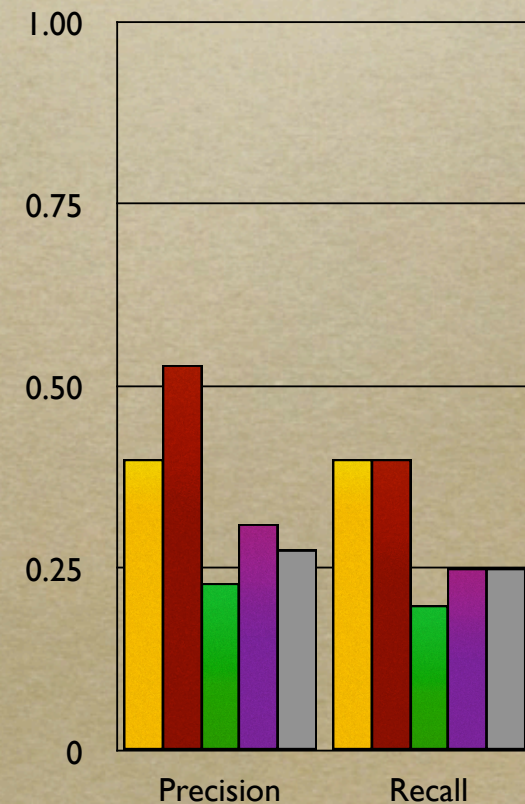
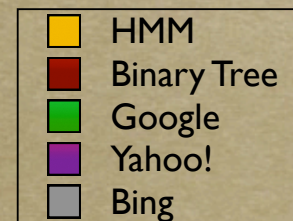

Experiments and results (cont..)

4. Experiment 4: Precision and recall

▶ Aim: To evaluate the correctness of the outputs

▶ Results:

	<i>HMM</i>	<i>Binary Tree</i>	<i>Google</i>	<i>Yahoo!</i>	<i>Bing</i>
<i>Precision</i>	0.4	0.53	0.23	0.3125	0.2777
<i>Recall</i>	0.4	0.4	0.2	0.25	0.25



Experiments and results (cont..)

4. Experiment 4: Precision and recall

▶ Description:

- Precision =
$$\frac{|\{\text{relevant results}\} \cap \{\text{retrieved results}\}|}{|\{\text{retrieved results}\}|}$$

- Recall =
$$\frac{|\{\text{relevant results}\} \cap \{\text{retrieved results}\}|}{|\{\text{relevant results}\}|}$$

Experiments and results (cont..)

4. Experiment 4: Precision and recall

▶ Description:

- Two lettered string experiment for calculating precision and recall
- 20 strings of length two are given to Japanese Professor and native Japanese friend
- They provided us most frequently used words for the given 20 strings
- This is our measure for calculating precision and recall values
- Check if suggestions given by HMM and binary tree and search engines match with the strings provided by humans

Conclusion

- Difficulties
 - Handling large number of observations
 - Randomly generating initial probability matrix
 - Japanese character charset issues
- Precision and recall
 - N-gram approach gives good results as compared to HMM
- Future work
 - Recognition of all different kanji symbols

References

1. [1996] Statistical Language Learning. Eugene Charniak. MIT Press. 1996.
2. The Tanaka Corpus. Retrieved November 23, 2010, from <http://www.csse.monash.edu.au/~jwb/tanakacorpus.html>
3. Rie Kubota Ando, Lillian Lee, Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences. Retrieved November 23, 2010, from <http://www.cs.cornell.edu/home/llee/papers/segmentjnle.pdf>
4. <http://en.wikipedia.org/wiki/File:Recall-precision.svg>

ありがとうございました。

