

JAPANESE KANJI SUGGESTION TOOL

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by

Sujata Dongre

December 2009

© 2009

Sujata Dongre

ALL RIGHTS RESERVED

ABSTRACT

JAPANESE KANJI SUGGESTION TOOL

by Sujata Dongre

Many times, we can see that if we enter a misspelled search term in any of the search engines like Google, it will provide some help with "Did you mean:...." In my project, I am trying to provide some suggestions for the wrong Japanese text entered by a user.

The Japanese language has three types of writing styles - Hiragana, Katakana and the most difficult, Kanjis. In old days Japanese script was only written vertically. However, the horizontal writing style is more common nowadays. A single Kanji may be used to write one or more different compound words. From the point of view of the reader, Kanjis are said to have one or more different "readings". Hence sometimes it becomes very difficult to understand what the Kanjis are and how to read them even if you know the Japanese language. There are various different translation tools available nowadays that provide translation help. However none of them provides any good suggestions for the incorrect Japanese words. In my project, I am developing a tool that will help users to correct their Japanese words in searching by giving them a list of correct Japanese words they might be looking for.

Table of Contents

1. Introduction
2. Deliverable 1: Search Japanese characters in Tanaka Corpus
3. Deliverable 2: Report on Hidden Markov Model and parsing algorithms
4. Deliverable 3: Programs for parsing algorithms
5. Deliverable 4: MySQL N-gram parser plugin for Japanese language
6. Conclusion
7. Future Work
8. References

List of Figures

1. Example of HMM
2. Guessed HMM
3. Example of running Viterbi.java
4. Example of running Forward.java
5. Example of running Backward.java
6. MySQL Show Plugins records
7. MySQL N-gram parser usage part1
8. MySQL N-gram parser usage part2
9. MySQL N-gram parser usage part3

List of Tables

1. Forward probabilities for “eye drops off shelf”
2. Backward probabilities for “eye drops off shelf”
3. New probabilities for “eye drops off shelf”

1. Introduction

This report provides detailed information about the work done in CS297: Preparation for Writing Project or Thesis course, for the project titled “Japanese Kanji Suggestion Tool”. In this report, you can see all the deliverables that were prepared for the different experiments and studies that were carried out.

In this multilingual world, we all come across different words in different languages. There are various websites where we can simply type the word and get its meaning in English. But the main difference between English and Japanese is that Japanese is written without any spaces in between the words. Hence, sometimes it becomes very hard to understand the right word/kanji even if you know Japanese. Consider following example:

You are reading some Japanese text on a website. You come across a sentence like "まず は、正しい英語学習法に頭をCHANGEしてください。" You do not understand the meaning of the sentence as you are unable to read the Kanjis. Now, you decide to use one of the translation tools such as Google, Yahoo Babel, or ALC. But you do not even understand which Kanjis to copy, and mistakenly, you select "習法". Search results given by the above three translation websites are as follows:

Yahoo: Learning Method

Google: 習法

ALC: Result not found.

Hence, the purpose of my project is to ask the user, "もしかして(Did you mean): "学習法"?", which is the correct Japanese term and also has the equivalent English meaning.

Even if you visit websites like Google Japan(<http://www.google.co.jp/>) or Yahoo Japan (<http://www.yahoo.co.jp/>) and search on these websites with some incorrect Japanese words, overall the suggestions given by these websites are not very good for the Japanese language. The purpose of my project is to develop a tool that will help Japanese language learners to read Japanese documents more efficiently.

2. Deliverable 1: Search Japanese characters in the Tanaka Corpus

Description:

In this deliverable, I studied different Japanese text corpora. In particular, I looked at the Kyoto Text Corpus and the Tanaka Corpus. The Tanaka Corpus consists of thousands of parallel Japanese-English sentences. The Tanaka Corpus was compiled by Professor Yasuhito Tanaka and his students at Hyogo University. For more detailed information about the Tanaka Corpus, please visit the following website:

<http://www.csse.monash.edu.au/~jwb/tanakacorporus.html>

Goal:

The goal of this deliverable is to write a program that will check whether the given input Japanese character/Kanji is present in the Tanaka Corpus file or not. If the character is present, the program will display all the lines containing that Japanese character. The name of this program is 'kanji_character_search.py'

Implementation and Results:

Input: Japanese character/Kanji

Output: Lines containing entered Japanese character/Kanji

Instructions to run the program:

- Python version 2.6.1
- Tanaka Corpus File
- `./kanji_character_search.py -f Tanaka Corpus filename -k Japanese character OR Kanji`

To run the program, you should provide two arguments, the filename for search, and the Japanese character/Kanji to be searched in the file. If any of these arguments is missing, the program will display a help message asking the user to enter correct arguments.

The Tanaka Corpus file includes Japanese as well as English sentences. To extract only the Japanese parsing sentences from the existing Tanaka Corpus file, there is one more

program. This program extracts Japanese parsing sentences from existing the Tanaka Corpus file and writes them to the new file. This new file will only have Japanese sentences. The name of this program is 'remove_english.py'

Example of running kanji_character_search.py:

- Woody:CS297 sujata\$./kanji_character_search.py -f test_corpus.txt -k ス
「17歳の時スクーター船で地中海を航海したわ」彼女はゆっくと注意深く言う。
「1秒6ペンスだからね」とボブが念を押す。
「4ポンド50ペンス」とボブが言う。

In the above example, the Tanaka corpus file is 'test_corpus.txt' and the Japanese character to be searched in the file is 'ス'. The output is all the lines having character 'ス' in it.

- Woody:CS297 sujata\$./kanji_character_search.py -f test_corpus.txt -k 日
Character not found.

If the character is not found in the file, then the message "Character not found." will be shown to the user.

- Woody:CS297 sujata\$./kanji_character_search.py -h
Usage: kanji_character_search.py [options]
Options:
-h, --help show this help message and exit
-f FILE, --file=FILE corpus file FILE
-k KANJI, --kanji=KANJI
kanji character to search in corpus

Help is available to the user for understanding the arguments.

- Woody:CS297 sujata\$./kanji_character_search.py
Please specify a corpus file. See -h for help.

An error message is displayed to the user if the user does not enter the corpus filename.

- Woody:CS297 sujata\$./kanji_character_search.py -f test_corpus.txt
Please specify a kanji/character to search. See -h for help.

An error message is displayed to the user if the user does not enter the Japanese character/Kanji for searching.

3. Deliverable 2: Report on the Hidden Markov Model and parsing algorithms

Description:

In this deliverable, I have studied the Hidden Markov Model and different algorithms that can be used for parsing. I have the working knowledge of Hidden Markov Model. Also, I understand how the algorithms such as Viterbi, Forward and Backward and HMM training work.

Goal:

This deliverable helped me to understand the different parsing technique that can be used for Japanese text. There are two parsing techniques, one is HMM and the other is Conditional Random Field. In this project, I am focusing on the HMM technique, is the most common technique used for parsing. The aim of this deliverable is to understand different HMM algorithms that can be programmed later for calculating probabilities and analyzing the Japanese text entered by the user.

Implementation and Results:

Hidden Markov Model

Introduction:

There are different Japanese text parsers available, two of them are the Chasen morphological analyzer and the MeCab parser. The Chasen morphological analyzer is based on the Hidden Markov Model while the MeCab parser is based on the Conditional Random Field. In my project, I will need a parser that will separate the Japanese text/kanjis entered by the user. Japanese texts are different from English texts in that, there are no word boundaries in Japanese texts. Still, the parsing techniques for Japanese text are also dependent on the Hidden Markov Model. In this report, I am going to explain what HMM is, why it is used, working of the HMM and the Viterbi algorithm giving my example.

What HMM is?

- The Hidden Markov Model is based on the Markov model. The Markov Model is a finite automata model which is used to find out what will be the next state depending on the current state. Hence to predict the future, you do not have to look at the past history of states of the model.

e.g. Consider the sentence “The monster swallowed _____”. In this sentence, after the verb swallowed, it is less likely that the next word will be a verb again. Hence by looking at the current word which is say “swallowed”, we can say that next word will be the most likely some article such as “a”, followed by noun such as “boy”. The purpose of Markov Model is to find the next word with the highest probability. The transition probability is assigned to each path. It is better to follow the path of traversing from one state to the other with the highest probability.

- The Markov Models are basically used to model sequence of events. These sequence of events could be fixed or not. The deterministic flow of events can be traffic lights turning from Green to Yellow to Red. The non-deterministic flow of events can be analyzing weather conditions.
- In the Markov Model, the output i.e., sequence of events/observations is simply the sequence of states visited.
- In the HMM, the states that the model is passed through is unknown to us. We only know the observation sequence, probabilities of transitioning from one state to other, the probability that for the current state what are the chances that given observation state will be generated.
- Hence, in the HMM, our purpose is to find the most likely sequence of states which could yield the observed sequence of symbols.

Why it is used?

- The HMM can be used in various applications such as speech recognition, part-of-speech tagging etc.
- I will explain POS (Part-Of-Speech) tagging with the HMM. In English, there are different types of POS tags such as DT(determiner), N(noun), V(verb) etc. Consider the sentence: The chocolate is sweet.

Here, the => DT,
 chocolate => N
 is => V
 sweet => ADJ

We would like to find out what is more likely the tag sequence. We want to find out the best tag sequence. The HMM will be used to find the sequence of states that will lead to the highest probability of tag t for given word sequence w . Hence, as per the Markov assumption, the word w_i depends on tag t_i and tag t_i depends on tag t_{i-1}

How the HMM works?:

- In the HMM, we know what the observable sequence is but we do not know what states it is been travelled to get this sequence.

- In the HMM, the states are not observable. We know some probabilities that the model will pass through.
- The HMM works as any other finite state machine having a set of hidden states, observable states, transition probabilities, initial state probabilities and output probabilities. The current state is not observable but each state produces an output with a certain probability.
- The HMM consists of a set of states as in any other finite automata model. Say states $\{q, r\}$, transition probabilities i.e., traveling from the state q to r for the given observation state, a set of observations such as string "aabb" and the probability of getting a symbol "b" for the current state say "r"
- The problem with the HMM is how to find out the state sequence that best explains the given set of observations? Because it is possible to have more than one state sequences that will lead to the same observation sequence. This problem could be solved with the use of the Viterbi algorithm.
- The working of the Viterbi algorithm is based on the dynamic programming approach. In the dynamic programming, the problem is divided into subproblems and the after solving each subproblem, the algorithm saves its answer in the table that could be used to solve the actual problem. The dynamic programming algorithm is used for solving problems having more than one possible solutions. The dynamic programming helps to find out the optimal solution for the given problem.
- Similarly in the Viterbi algorithm, the purpose is to find the best path sequence that will lead to the given observation states.
- My example for the Viterbi algorithm for given HMM is as follows:
Sentence: Eye drops off shelf.
As per the HMM,
Set of states = $\{q, r\}$
Set of observations = $\{\text{eye, drops, off, shelf}\}$

As the Viterbi algorithm follows the dynamic programming approach, it is required to store the best path and its probability for each possible state.

HMM Training

We have seen that what the HMM is, why it is used and how it works. Basically there are three problems with the given Hidden Markov Model.

They are as follows:

1. Given the Hidden Markov Model and a sequence of observations, we would like to know what is the sequence of hidden states that leads to the given set of observations? This problem as described above can be solved using the Viterbi algorithm.
2. Given the Hidden Markov Model as well as emission and transition probabilities, what will be the probability of a given sequence of observations? This problem can be solved using the Forward algorithm. The Forward algorithm is similar to the Viterbi algorithm except the purpose of the Forward algorithm is find the sum of all possible ways to get to some end state. Thus, in the Forward algorithm, the probability of reaching to some end state is the total of product of all the probabilities of paths leading to that end state.

In simple words,

Forward Probability at timestep $(t + 1)$ = Sum of (Forward Probability at timestep t * transition probability that will end up in that state)

Consider the above example of the Viterbi algorithm.

Time ticks	1	2	3	4	5
input	e	eye	eye drops	eye drops off	eye drops off shelf
Forward Probability of state q at timestamp t	1.0	0.28	0.0876	0.0149	0.0023
Forward Probability of state r at timestamp t	0.0	0.12	0.0444	0.007	0.0012
Total Probability	1.0	0.4	0.132	0.0219	0.0035

Table 1: Forward probabilities for “eye drops off shelf”

As explained earlier, given the observation “eye”, what is the total probability i.e., sum of the probabilities of all paths that one will end up in state “q” or in state “r”

Hence, in above example,

forward probability of state q at time 2

= (forward probability of state q at time 1 * transition probability from q -> q * emission probability for observation eye from state q)

+ (forward probability of state r at time 1 * transition probability from r -> q * emission probability for observation eye from state r)

$$= (1.0 * 0.7 * 0.4) + (0.0 * 0.6 * 0.2)$$

$$= 0.28$$

Similarly, we can calculate the forward probability for other states and at different timestamps. As we are moving forward in the sequence, this is called as forward probabilities.

We can also use the Backward algorithm instead of the Forward algorithm. The difference between the Backward algorithm and the Forward algorithm is, in the Backward algorithm we calculate the probability of sequence by working backward.

Time ticks	1	2	3	4	5
Input	eye drops off shelf	drops off shelf	off shelf	shelf	ε
Backward probability of state q at timestamp t	0.0035	0.0083	0.032	0.1	1
Backward probability of state r at timestamp t		0.0105	0.018	0.3	1

Table 2: Backward probabilities for “eye drops off shelf”

Hence, in the above example,

the backward probability of state q at time 2

= (backward probability of state q at time 3 * transition probability from q -> q *
 emission probability for observation drops at state q)
 + (backward probability of state r at time 3 * transition probability from q -> r * emission
 probability for observation drops at state q)

$$= (0.032 * 0.7 * 0.3) + (0.018 * 0.3 * 0.3)$$

$$= 0.0083$$

3. Given an output sequence O, what HMM transition probabilities maximize the likelihood of the sequence? i.e We have the model and a set of all observations, how can we adjust the parameters to increase the probability of the observations given the model? This problem can be solved using the Forward-Backward algorithm, also known as the Baum-Welch algorithm.

Using an initial parameter instantiation, the Forward-Backward algorithm iteratively re-estimates the parameters and improves the probability that given observations are generated by the new parameters. So, we need to re-estimate three parameters:

- initial state distribution
- transition probabilities
- emission probabilities

How HMM training works?

- The purpose of using the HMM training is to find the HMM that maximizes the probability of the given observation sequence. We know the observation sequence, we have to find the three parameters as described above.
- To achieve this we will use the Forward-Backward/Baum-Welch algorithm. We follow the hill climbing approach. So we start with some random HMM model and iteratively make small changes to the solution, improving it a little each time. When there is no more improvement possible, the algorithm terminates.
- So we do not know what the model is. We can work out the probability of the observation sequence using some randomly chosen model. By looking at it, we can see which state transitions and emissions were probably used the most. By increasing the probability of those, we can choose a revised better model which gives a higher probability to the observation sequence. Thus, we are following the process of maximization which is referred to as training the model.
- Hence, we will guess randomly transition probabilities from state q to state r. But then how can we find the new re-estimated probability?

New probability from state q to r = (forward probability from state q at timestamp t *
 backward probability from state r at timestamp t+1 *
 transition probability from state q to state r for the given
 observation)

- We will use this newly calculated probability for re-estimating start probabilities, transition probabilities and emission probabilities. Consider following diagram for guessed HMM:

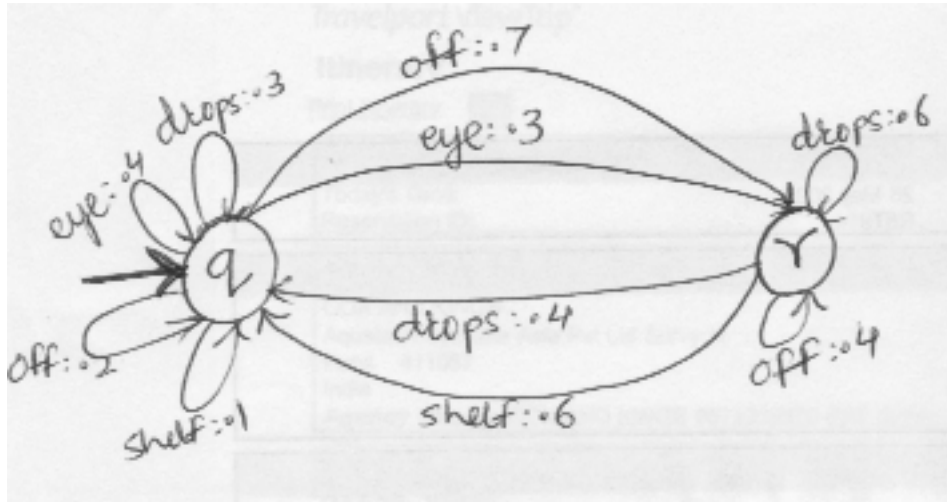


Figure 2: Guessed HMM

Time ticks	t1	t2	t3	t4		
Input	eye	drops	off	shelf	Total P	New P
q -> r for observation eye	0.003	0	0	0	0.003	0.23
r -> r for observation drops	0	0.001	0	0	0.001	0.08
r -> r for observation off	0	0	0.005	0	0.005	0.38

Time ticks	t1	t2	t3	t4		
r -> q for observation shelf	0	0	0	0.004	0.004	0.3
					0.013	

Table 3: New probabilities for “eye drops off shelf”

Thus, in the above example,

Total probability from state q to state r for observation eye =

forward probability of state q at time t1 *

transition probability from state q to state r for observation ‘eye’ *

backward probability of state r at time t2

$$= 1.0 * 0.3 * 0.0105$$

$$= 0.00315$$

$$\approx 0.003$$

Similarly, we can calculate the new probability for other observations as well. After calculating all the total probabilities for remaining transitions, we calculate its total i.e. Summation(Total P). This is nothing but the expected number of transitions from one state to the other state.

New probability from state q to state r =

Total P from state q to state r / Summation (Total P)

$$= 0.003 / 0.013$$

$$= 0.23$$

Our purpose of calculating total probabilities is we would like to find out the new probability that will be greater than the previous probability. If we think that the new probability and old probability are somewhat similar, we will stop iterating through the HMM and that will be our new re-estimated parameters.

4. Deliverable 3: Programs for parsing algorithms

Description:

In the deliverable 2, I studied different HMM algorithms that can be used for parsing. In this deliverable, I developed programs for Viterbi algorithm, Forward and Backward algorithms.

Goal:

In my tool, when the user enters the Japanese characters, I have to split the Japanese characters to understand exactly what the user is looking for. I need to parse the Japanese text using some algorithms.

Implementation and Results:

Output of Viterbi Program 'Viterbi.java': The most probable path and the probability of that path

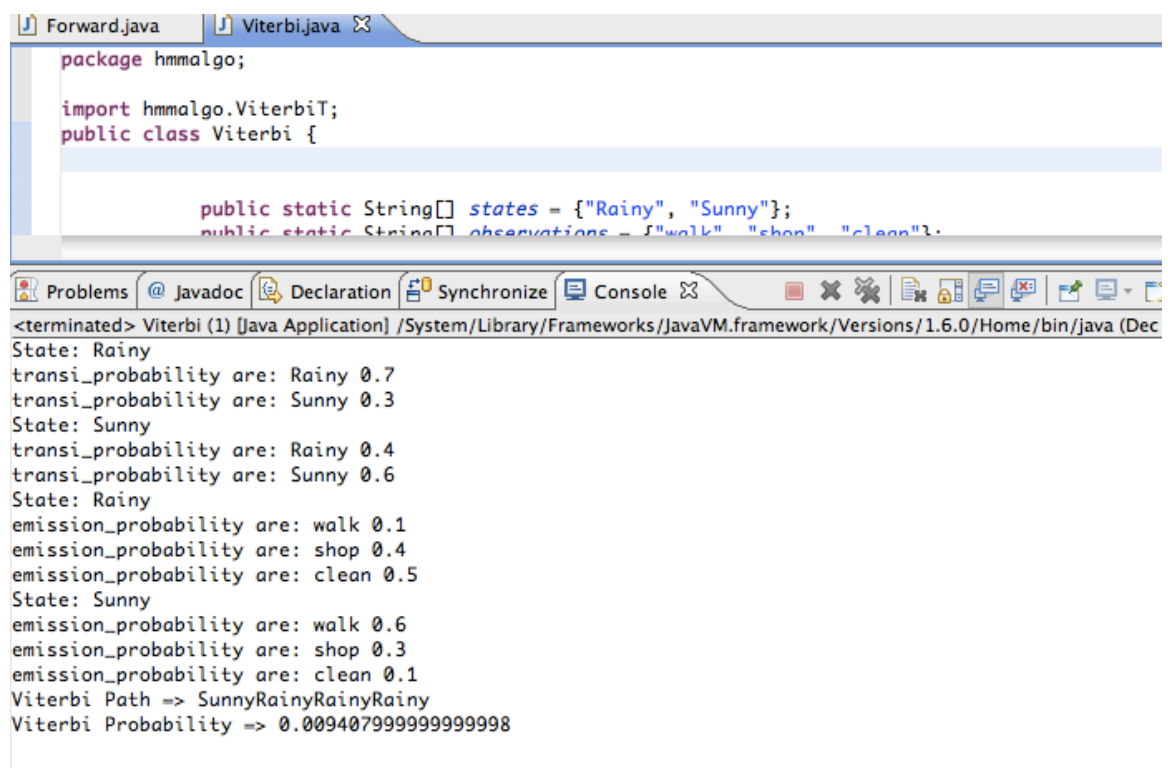
Output of Forward Program 'Forward.java': Forward probability

Output of Backward Program 'Backward.java': Backward probability

Instructions to run the program:

- JDK 1.5
- javac hmmalgo/Viterbi.java
- java hmmalgo/Viterbi
- javac hmmalgo/Forward.java
- java hmmalgo/Forward
- javac hmmalgo/Backward.java
- java hmmalgo/Backward

IDE used: Eclipse Ganymede



The screenshot shows an IDE with two tabs: 'Forward.java' and 'Viterbi.java'. The 'Viterbi.java' tab is active, displaying the following code:

```
package hmmalgo;

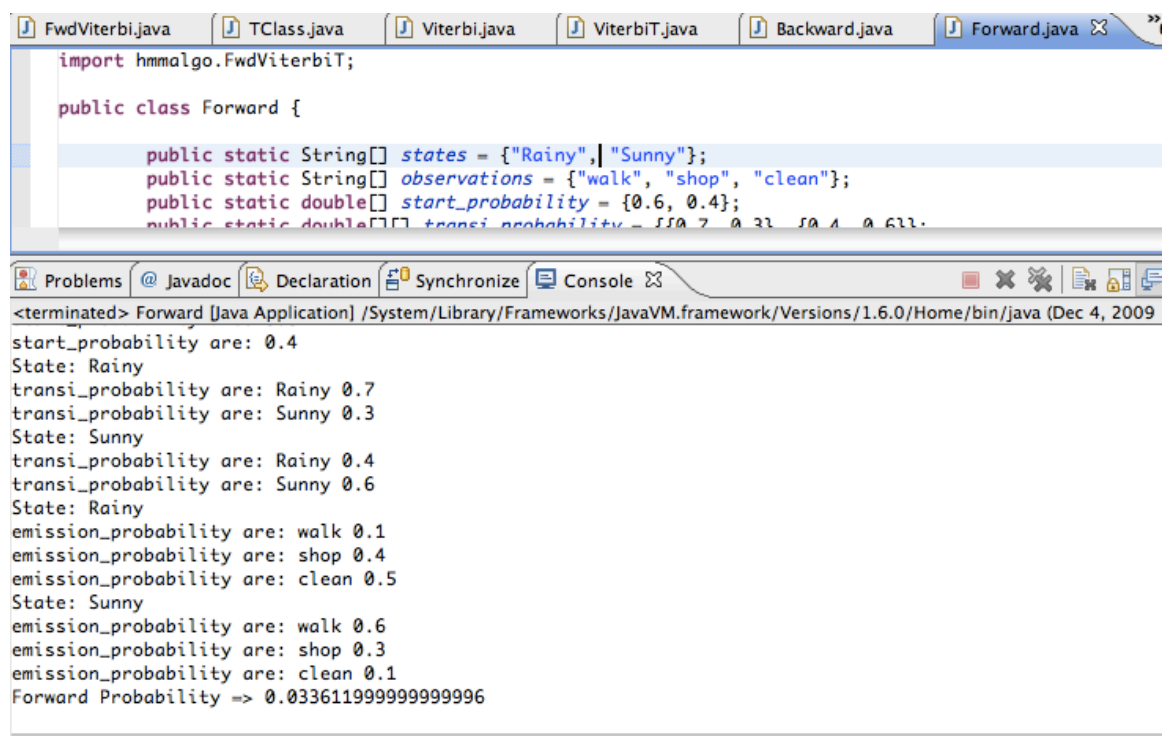
import hmmalgo.ViterbiT;
public class Viterbi {

    public static String[] states = {"Rainy", "Sunny"};
    public static String[] observations = {"walk", "shop", "clean"};
```

Below the code editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> Viterbi (1) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (Dec
State: Rainy
transi_probability are: Rainy 0.7
transi_probability are: Sunny 0.3
State: Sunny
transi_probability are: Rainy 0.4
transi_probability are: Sunny 0.6
State: Rainy
emission_probability are: walk 0.1
emission_probability are: shop 0.4
emission_probability are: clean 0.5
State: Sunny
emission_probability are: walk 0.6
emission_probability are: shop 0.3
emission_probability are: clean 0.1
Viterbi Path => SunnyRainyRainyRainy
Viterbi Probability => 0.009407999999999998
```

Figure 3: Example of running Viterbi.java



The screenshot shows an IDE with several tabs: FwdViterbi.java, TClass.java, Viterbi.java, ViterbiT.java, Backward.java, and Forward.java. The Forward.java tab is active, displaying the following code:

```
import hmmalgo.FwdViterbiT;

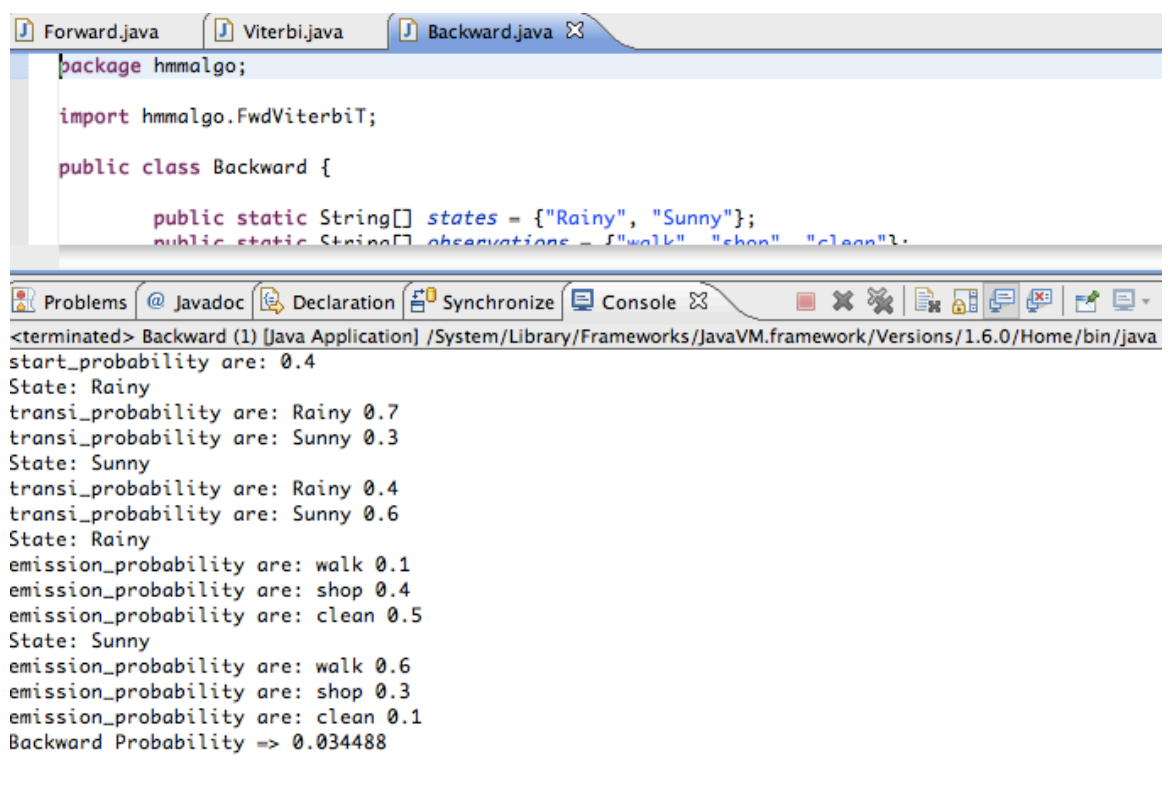
public class Forward {

    public static String[] states = {"Rainy", "Sunny"};
    public static String[] observations = {"walk", "shop", "clean"};
    public static double[] start_probability = {0.6, 0.4};
    public static double[][] transi_probability = {{0.7, 0.3}, {0.4, 0.6}};
```

Below the code editor, the console window shows the output of the program:

```
<terminated> Forward [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java (Dec 4, 2009)
start_probability are: 0.4
State: Rainy
transi_probability are: Rainy 0.7
transi_probability are: Sunny 0.3
State: Sunny
transi_probability are: Rainy 0.4
transi_probability are: Sunny 0.6
State: Rainy
emission_probability are: walk 0.1
emission_probability are: shop 0.4
emission_probability are: clean 0.5
State: Sunny
emission_probability are: walk 0.6
emission_probability are: shop 0.3
emission_probability are: clean 0.1
Forward Probability => 0.033611999999999996
```

Figure 4: Example of running Forward.java



The screenshot shows an IDE with three tabs: Forward.java, Viterbi.java, and Backward.java. The Backward.java tab is active, displaying the following code:

```
package hmmalgo;

import hmmalgo.FwdViterbiT;

public class Backward {

    public static String[] states = {"Rainy", "Sunny"};
    public static String[] observations = {"walk", "shop", "clean"};
```

Below the code editor, the console window shows the output of the program:

```
<terminated> Backward (1) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java
start_probability are: 0.4
State: Rainy
transi_probability are: Rainy 0.7
transi_probability are: Sunny 0.3
State: Sunny
transi_probability are: Rainy 0.4
transi_probability are: Sunny 0.6
State: Rainy
emission_probability are: walk 0.1
emission_probability are: shop 0.4
emission_probability are: clean 0.5
State: Sunny
emission_probability are: walk 0.6
emission_probability are: shop 0.3
emission_probability are: clean 0.1
Backward Probability => 0.034488
```

Figure 5: Example of running Backward.java

5. Deliverable 4: MySQL N-gram parser plugin for Japanese

Description:

The MySQL full-text search is used for faster search through large databases. This feature was available for only the English language before the MySQL 5.1 version. From the MySQL 5.1, the N-gram plugin can be added for using the full-text search functionality on languages like Chinese, Japanese and Korean.

Goal:

In this deliverable 4, I am experimenting with the N-gram plugin that can be added to the existing plugins in the MySQL. After the successful installation of the plugin, I can run the full-text search for the Japanese language.

Implementation and Results:

Instructions for compiling and installing the N-gram plugin:

- Download bigram plugin from: <http://sourceforge.net/projects/mysqlftppc/>
- Once you download the tar file, extract it under some folder
- Now run configure script. Use following command:
Woody:mysqlftppc-bigram-1.6 sujata\$./configure CFLAGS='-arch i686' LDFLAGS='-arch i686' CC='gcc -m32' CXX='g++ -m32' CHOST='i686-apple-darwin10.0.0' --target='i686-apple-darwin10.0.0' --build='i686-apple-darwin10.0.0' --host='i686-apple-darwin10.0.0' --with-mysql-config=/Applications/XAMPP/xamppfiles/bin/mysql_config
- Once you are done with the compilation, run following commands for installation:
Woody:mysqlftppc-bigram-1.6 sujata\$ make CFLAGS='-arch i686' LDFLAGS='-arch i686' CC='gcc -m32' CXX='g++ -m32'
Woody:mysqlftppc-bigram-1.6 sujata\$ sudo make install
- This will install the plugin libraries in /Applications/XAMPP/xamppfiles/lib/mysql/plugin. You will find the file with the name as 'libftbigram.so' under this folder.

- Once you are done with the installation, open my.cnf file from /Applications/XAMPP/xamppfiles/etc

- Search for '[mysqld]' in my.cnf file and add following lines in my.cnf file:

The MySQL server

[mysqld]

ft_min_word_len = 1

plugin_dir = /Applications/XAMPP/xamppfiles/lib/mysql/plugin

- Restart the MySQL server since my.cnf file is modified.
- After restarting the MySQL server, connect to the MySQL server by following command:

Woody:mysqlftppc-bigram-1.6 sujata\$ mysql -uroot -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 8

Server version: 5.1.37 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

- Now, type the command for installing the plugin:
mysql>INSTALL PLUGIN bigram SONAME 'libftbigram.so'
- To check whether plugin is added successfully or not, you can use the following command:
mysql> SHOW PLUGINS;

You should get the records as follows:

```
mysql> show plugins;
+-----+-----+-----+-----+-----+-----+
| Name | Status | Type | Library | License |
+-----+-----+-----+-----+-----+
| binlog | ACTIVE | STORAGE ENGINE | NULL | GPL |
| ARCHIVE | ACTIVE | STORAGE ENGINE | NULL | GPL |
| CSV | ACTIVE | STORAGE ENGINE | NULL | GPL |
| FEDERATED | DISABLED | STORAGE ENGINE | NULL | GPL |
| MEMORY | ACTIVE | STORAGE ENGINE | NULL | GPL |
| InnoDB | ACTIVE | STORAGE ENGINE | NULL | GPL |
| MyISAM | ACTIVE | STORAGE ENGINE | NULL | GPL |
| MRG_MYISAM | ACTIVE | STORAGE ENGINE | NULL | GPL |
| bigram | ACTIVE | FTPARSER | libftbigram.so | BSD |
+-----+-----+-----+-----+-----+
9 rows in set (0.12 sec)
```

Figure 6: MySQL Show Plugins records

Usage:

```
mysql> use test;
Database changed
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| del4            |
| t               |
| tbl             |
| tbl1            |
+-----+
5 rows in set (0.00 sec)

mysql> select * from del4;
Empty set (0.00 sec)

mysql> insert into del4 values('=====');
Query OK, 1 row affected (0.00 sec)

mysql> select * from del4;
+-----+
| body |
+-----+
| にほん |
+-----+
1 row in set (0.00 sec)

mysql> insert into del4 values('=====');
Query OK, 1 row affected (0.00 sec)

mysql> select * from del4;
+-----+
| body |
+-----+
| にほん |
| 日本語 |
+-----+
2 rows in set (0.01 sec)
```

Figure 7: MySQL N-gram parser usage part1

```
mysql> insert into del4 values('==');
Query OK, 1 row affected (0.01 sec)

mysql> select * from del4;
+-----+
| body |
+-----+
| にほん |
| 日本語 |
| 本 |
+-----+
3 rows in set (0.00 sec)

mysql> select * from del4 where match(body) against ('== ' IN BOOLEAN MODE);
+-----+
| body |
+-----+
| 日本語 |
| 本 |
+-----+
2 rows in set (0.00 sec)

mysql> select * from del4 where match(body) against ('本' IN BOOLEAN MODE);
+-----+
| body |
+-----+
| 日本語 |
| 本 |
+-----+
2 rows in set (0.00 sec)

mysql> select * from del4 where match(body) against ('== ' IN BOOLEAN MODE);
+-----+
| body |
+-----+
| にほん |
+-----+
1 row in set (0.00 sec)
```

Figure 8: MySQL N-gram parser usage part2

```
mysql> select * from del4 where match(body) against ('ほ ' IN BOOLEAN MODE);
+-----+
| body |
+-----+
| にほん |
+-----+
1 row in set (0.00 sec)
```

Figure 9: MySQL N-gram parser usage part3

6. Conclusion

In CS297, I learned about different algorithms and parsing techniques that are available for the Japanese language. I understood the HMM in great details.

The deliverable 1 program can be used to for searching the Japanese word in the Japanese text corpora. The deliverable 2 is all about understanding the HMM and the HMM training algorithm. The deliverable 2 will help me in writing parsing programs for the Japanese text. In the deliverable 3, I developed programs that can be used later for the HMM training algorithm. The deliverable 4 which is the MySQL N-gram plugin can be used for the full-text search functionality in the Japanese language.

7. Future Work

In 2010, I will extend the present project to develop a tool that will provide a list of similar Japanese words for the wrong Japanese text entered by the user. I will develop a software tool, that will take the Japanese text as an input and produce a list of correct Japanese words.

The deliverables that I worked in CS297 are the foundation for the development of my final project.

8. References

- [1] Viterbi Algorithm. Retrieved November 25, 2009, from http://en.wikipedia.org/wiki/Viterbi_algorithm
- [2] MySQL full-text parser collection. Retrieved November 25, 2009, from http://sourceforge.net/apps/mediawiki/mysqlftppc/index.php?title=Main_Page
- [3] MySQL full-text search capabilities. Retrieved November 25, 2009, from <http://www.devarticles.com/c/a/MySQL/Getting-Started-With-MySQLs-Full-Text-Search-Capabilities/1/>
- [4] Compiling i386 dynamic library. Retrieved November 25, 2009, from <http://discussions.apple.com/thread.jspa?messageID=10497361>