Machine Discovery Homework 1-1

Student Name and ID

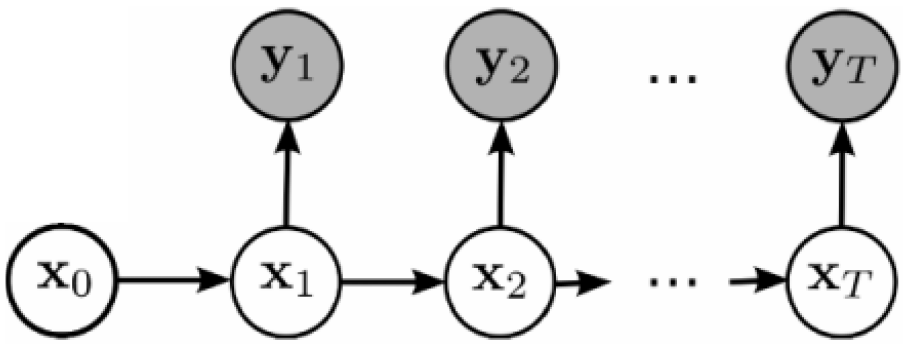
- 簡瑋德
- B03902015

Description

- Given Bigram Language Model and Encoding Table, design a model to decode a text file

Framework

- Architecture
    - Seperate each line into words by a space
    - Use Viterbi Algorithm to predict the word
    - The spelling of the predicted word is corrected by Jazzy, which is a Java-based spell-checker
- Assumption
    - Bigram Language Model: $P(w_1, w_2, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})$
    - Probabilistic Encoding Function
- Probalilistic Graphical Model



    - $\forall x_i, y_i \in X$, where $X = \{$ Lower-case Alphabats $\} \bigcup \{$ Number 0 to 9 $\} \bigcup \{$ space $\}$
    - x_0 is the random variable denoting the symbol in front of the word, and $x_1, x_2, ..., x_T$ are the random variables of predicted symbols within a word
    - $y_1, y_2, ..., y_T$ are the random variables of observed symbols within a word
    - $P(x_i|x_{i-1})$ and $P(y_i|x_i)$ are given
    - Define the objective function $J = P(x_1, x_2, ..., x_T, y_1, y_2, ..., y_T|x_0 = space)$
    - With the help of GM, we can decompose J to $\prod_{i=1}^{T} P(x_i|x_{i-1})P(y_i|x_i)$
- Viterbi Algorithm
    - Suppose sequence $P = \{p_1, p_2, ..., p_T\}$ are the predicted symbols of a word
    - Sequence $O = \{o_1, o_2, ..., o_T\}$ are the observed symbols of a word
    - Define $M(k, j, i) = P(x_t = X_j|x_{t-1} = X_k) \times P(y_t = o_i|x_t = X_j)$
    - We want to output the best $\{p_1, p_2, ..., p_T\}$ by recording the best case of $P_{idx,j}$ and $J_{idx,t}$ for $t = 1, 2, ..., T$ and $idx = 1, 2, 3, ..., size(X)$, where $P_{idx,j} = \{p'_1, p'_2, ..., p'_{idx-1}, X_{idx}\}$ such that $J_{idx,t} = \prod_{i=1}^{t} P(x_i = p'_i|x_{i-1} = p'_{i-1})P(y_i = o_i|x_i = p'_i)$ is the maximum

```
for each symbol in X
    J[idxOf(symbol), 1] = p(x1 = symbol | x0 = ' ')
    P[idxOf(symbol), 1] = 0
for i in {2, 3, ..., T}
    for each symbol in X
        J[idxOf(symbol), i] = max_k( J[k, i - 1] * M(k, j, i) )
        P[idxOf(symbol), i] = argmax_k( J[k, i - 1] * M(k, j, i) )
curIdx = argmax_idx( J[idx, T] )
predit = []
predit.push_front( X[curIdx] )
for i in {T, T - 1, ..., 2}
    curIdx = P[curIdx, i]
    predit.push_front( X[curIdx] )
return predit
```

Setings and Configuration

- `pred.txt` : The predicted result
- `used-tools.txt` : A list of third-part tools
- `report.pdf` : The report of the homework
- `README.txt` : Instructions to execute the program
- `src/` : Source codes
- `bin/` : Java compiled class files
- `doc/` : Documents
    - `docs/bigram.txt` : Text file of Bigram Language Model
    - `docs/dictionary.txt` : Dictionary for Jazzy spell-checker
    - `docs/encode.txt` : Text file of the Probabilistic Encoding Function
    - `docs/test.txt` : Test data for the homework
- `libs/` : Third-part libraries
- `Makefile` : Makefile for Linux
- Compile and Run:
    - Makefile is available
        - `B03902015$ make`
        - `B03902015$ make run`
    - Commands (Linux)
        - `B03902015$ javac -d bin -sourcepath src -cp libs/jazzy-core-0.5.2.jar src/launch/Main.java`
        - `B03902015$ java -Xmx1024M -cp bin:libs/jazzy-core-0.5.2.jar launch.Main`
    - Commands (Windows)
        - `B03902015$ javac -d bin -sourcepath src -cp libs/jazzy-core-0.5.2.jar src/launch/Main.java`
        - `B03902015$ java -Xmx1024M -cp bin;libs/jazzy-core-0.5.2.jar launch.Main`
    - The process will generate `docs/pred.txt` according to `docs/test.txt` and it takes about 30 minutes (90% of the time is cosumed by the spell-checker) and at least 800M RAM

References

- Viterbi Algorithm
- Jazzy