Algorithm 1 Best Path HMM

```
procedure TransitionProbability(obs)
    trans = numpy.array(9 X 9)
    sum_trans = numpy.array(9 X 1)
    trans_prob = numpy.array(9 X 9)
    for row in range(obs(all the rows)) do
        for col in range(obs(all the columns of each row)) do
            $\textbf{trans}[obs[row][col]][obs[row][col+1]]{+}{+}$
    for r in range(trans(all the rows)) do
        for c in range(trans(all the columns of each row)) do
            $\textbf{sum\_trans}[r] += \textbf{trans}[r][c]$
    for r in range(trans(all the rows)) do
        for c in range(trans(all the columns of each row)) do
            $\textbf{trans\_prob}[r][c] = \textbf{trans}[r][c]/\textbf{sum\_trans}[r]$
    return trans_prob
procedure EmissionProbability(obs)
    ems = numpy.array(9 X 37)
    sum_ems = numpy.array(1 X 37)
    ems_prob = numpy.array(9 X 37)
    for row in range(obs(all the rows)) do
        for col in range(obs(all the columns of each row)) do
            $\textbf{ems}[obs[row][col]][col]{+}{+}$
    for c in range(ems(all the columns)) do
        for r in range(ems(all the rows of each column)) do
            $\textbf{sum\_ems}[0][c] += \textbf{ems}[r][c]$
    for r in range(ems(all the rows)) do
        for c in range(ems(all the columns of each row)) do
            $\textbf{ems\_prob}[r][c] = \textbf{ems}[r][c]/\textbf{sum\_ems}[0][c]$
    return ems_prob
procedure AverageTimeProbability(obs, obs_time)
    ems_time = numpy.array(9 X 37)
    sum_ems_time = numpy.array(1 X 37)
    ems_time_prob = numpy.array(9 X 37)
    for row in range(obs(all the rows)) do
        for col in range(obs(all the columns of each row)) do
            $\textbf{ems\_time}[obs[row][col]][col] += \textbf{obs\_time}[r][c]$
    for r in range(ems_time(all the rows)) do
        for c in range(ems_time(all the columns of each row)) do
            $\textbf{sum\_ems\_time}[r] += \textbf{ems\_time}[r][c]$
    for r in range(ems_time(all the rows)) do
        for c in range(ems_time(all the columns of each row)) do
            $\textbf{ems\_time\_prob}[r][c] = \textbf{ems\_time}[r][c]/\textbf{sum\_ems\_time}[0][c]$
    return ems_time_prob
```

Algorithm 2 Main function of Best Path HMM

```
procedure Main
    obs1 = (Matrix of observations from file 1)
    obs2 = (Matrix of observations from file 2)
    obs3 = (Matrix of observations from file 3)
    obs4 = (Matrix of observations from file 4)
    obs_time1 = (Matrix of observations from times file 1)
    obs_time2 = (Matrix of observations from times file 2)
    obs_time3 = (Matrix of observations from times file 3)
    obs_time4 = (Matrix of observations from times file 4)
    obs = combine all the obs1,2,3,4
    obs_time = combine all the obs_time1,2,3,4
    trans_mat = function TransitionProbability(obs)
    ems_mat = function EmissionProbability(obs)
    ems_mat_time = function AverageTimeProbability(obs, obs_time)
    path = numpy.array to save best state sequence
    for t in range of(length of observation sequence row) do
        for s in range of(number of states (i.e 9)) do
            if $ems\_mat > 0.8$ then
                $path[t] = numpy.argmax(ems\_mat)$
            if $0.8 > ems\_mat > 0.1$ then
                $path[t] = numpy.argmax(ems\_mat + trans\_mat)$
            if $ems\_mat < 0.1$ then
                $path[t] = numpy.argmax(ems\_mat + trans\_mat + ems\_mat\_time)$
    return path
```

Algorithm 3 K-Fold Cross Validation

```
procedure CrossValidation(path1, path2)
    count = 0
    for pos in range of(length of observation sequence row) do
        if $path1[pos] \neq path2[pos]$ then
            count + +
    return count/37
```