ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

MÔN HỌC: THỐNG KÊ MÁY TÍNH VÀ ỨNG DỤNG

Báo cáo Project 1:

The Discrete HMM for Classification of Vietnamese Documents

Nhóm sinh viên

Nguyễn Huy Hải – 18120023

Phạm Công Minh – 18120058

Nguyễn Thanh Tùng – 18120104

Nguyễn Thị Hồng Nhung – 18120498

Mục lục

1 Phân công công việc và mức độ hoàn thành của mỗi thành viên nhóm

- **Nguyễn Huy Hải - 18120023**
  - Tổng hợp và viết báo cáo, phân tích về mô hình *Hidden Markov*.

  Đánh giá: 100%

- **Phạm Công Minh - 18120058**
  - Xây dựng và cải tiến mô hình *Hidden Markov* hoàn chỉnh.

  Đánh giá: 100%

- **Nguyễn Thanh Tùng - 18120104**
  - Thu thập, tiền xử lý và phân tích dữ liệu, xây dựng *Pipeline* hoàn chỉnh để thực hiện huấn luyện và thử nghiệm mô hình.

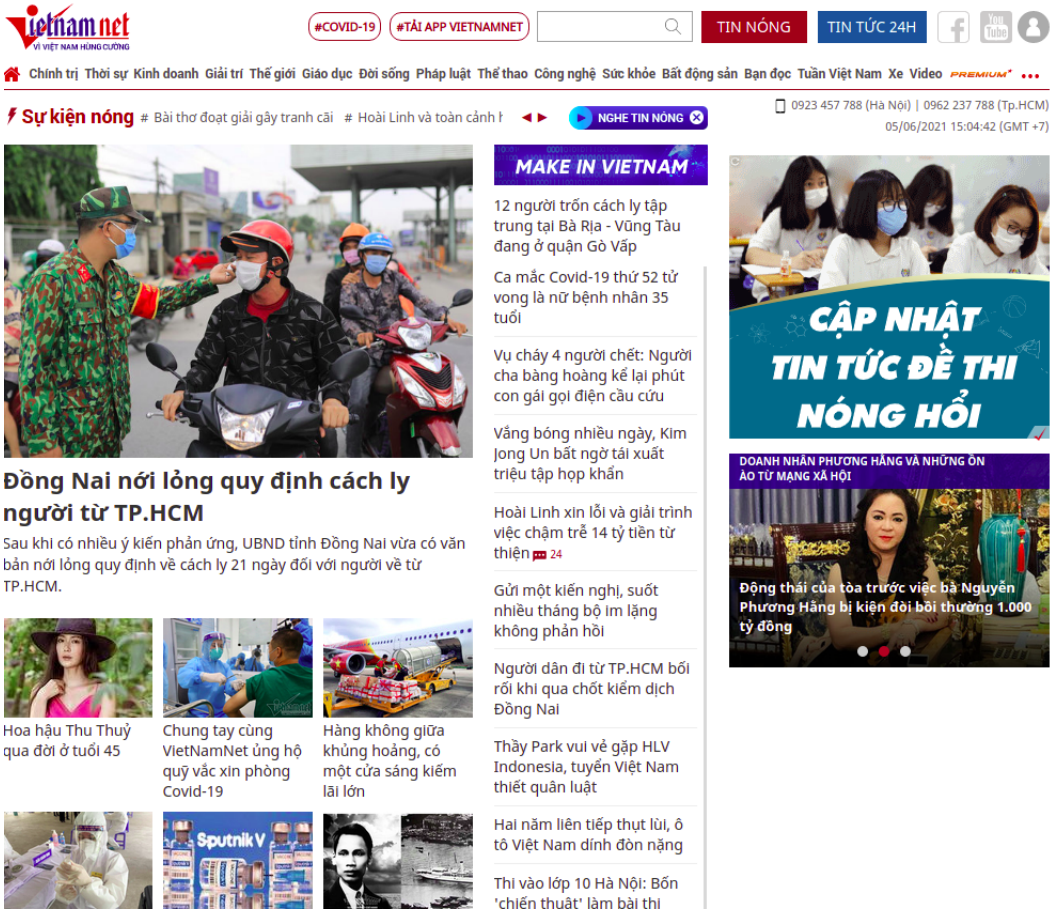
  Đánh giá: 100%

- **Nguyễn Thị Hồng Nhung - 18120498**
  - Tìm hiểu về các thuật toán có trong *Hidden Markov Model* cũng như cách xây dựng mô hình.

  Đánh giá: 100%

2 Bài toán đặt ra - Phân loại văn bản Tiếng Việt

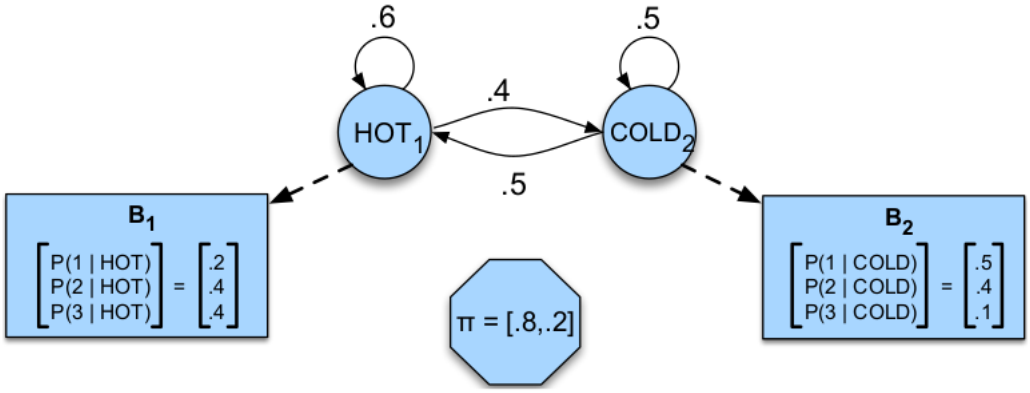
Trong thời đại công nghệ ngày càng phát triển như hiện nay, kéo theo nhu cầu tự động hóa ngày càng lớn hơn. Khối lượng dữ liệu, thông tin cũng ngày càng nhiều hơn, đòi hỏi cần tạo ra những công cụ để hỗ trợ phân loại văn bản, thông tin một cách nhanh chóng và hiệu quả.
Trong đồ án này, nhóm sẽ đi tìm hiểu xây dựng một *Hidden Markov Model (HMM)* để thực hiện việc phân loại văn bản Tiếng Việt. Kèm theo đó là đánh giá mức độ hiệu quả của các thuật toán của mô hình *HMM*, so sánh với một thuật toán quen thuộc khác cũng liên quan tới xác suất là *Naive Bayes*. Ở đây ta sẽ lấy dữ liệu từ trang báo **vietnamnet.vn** để làm tập *train* và **baomoi.com** để làm tập *test*, đảm bảo tính khách quan của mô hình.



Trước khi đi vào xây dựng một *HMM* hoàn chỉnh, trước tiên ta cùng đi tìm hiểu *HMM* là gì cũng như các thuật toán của nó.

3 Các thuật toán của Hidden Markov Model và ứng dụng mô hình vào bài toán đặt ra

Xét bài toán số cây kem (*Ice Cream*) được ăn (1, 2 hay 3) phụ thuộc vào thời tiết (*Hot* hoặc *Cold*). Với *Hot* (H) và *Cold* (C) là 2 trạng thái ẩn (*Hidden State*) và $O = \{1, 2, 3\}$ tương ứng là số cây kem được ăn.



Mô hình *Hidden Markov Model (HMM)* được xây dựng dựa vào 3 vấn đề cơ bản sau:

- Vấn đề 1 (*Likelihood*): Cho trước có $\lambda = (A, B)$ và chuỗi quan sát O, xác định $P(O|\lambda)$.
- Vấn đề 2 (*Decoding*): Cho trước chuỗi quan sát O và *HMM* có $\lambda = (A, B)$, xác định chuỗi trạng thái ẩn Q tốt nhất.
- Vấn đề 3 (*Learning*): Cho trước chuỗi quan sát O và tập hợp các trạng thái trong *HMM*, học *HMM* để tìm xác định 2 tham số A và B.

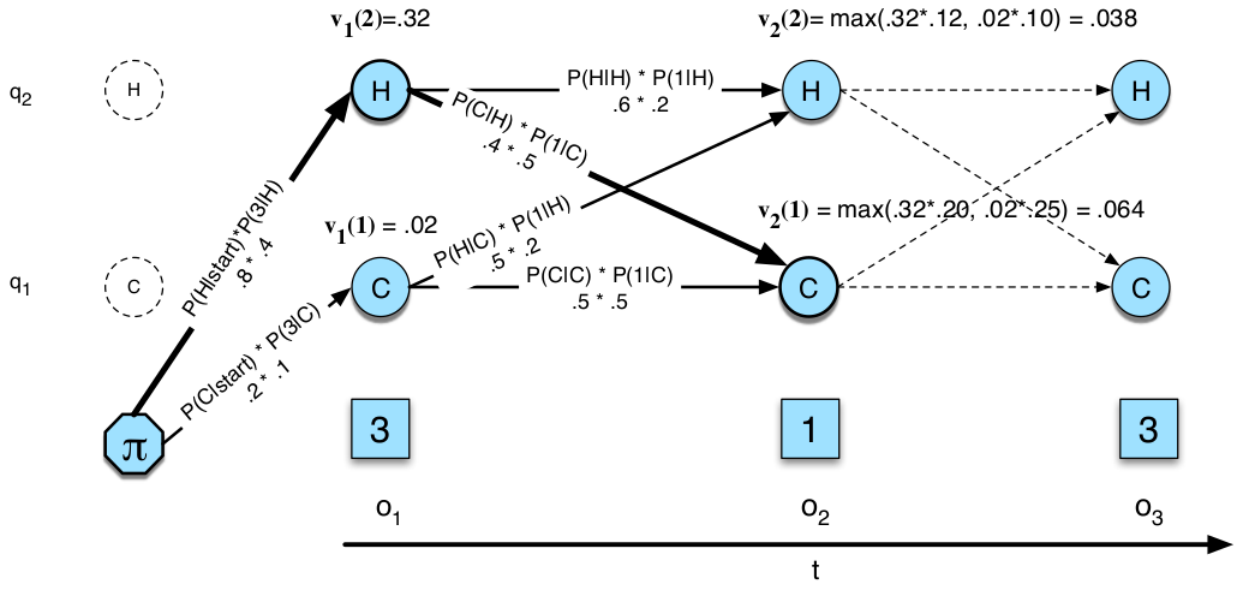
Ở đây ta chỉ quan tâm đến hai vấn đề 2 và 3!

3.1 Decoding: Thuật toán Viterbi

Decoding (Giải mã) là nhiệm vụ xác định chuỗi nguồn (chuỗi trạng thái ẩn) của một chuỗi quan sát cho trước.
Trong ví dụ về *Ice Cream* nêu trên, cho trước một chuỗi quan sát về số lượng cây kem được ăn là 3 1 3 và một *HMM*, nhiệm vụ của **decoder** là tìm ra chuỗi trạng thái ẩn (*hidden state*) tốt nhất tương ứng với chuỗi quan sát trên. Như đã nêu ở trên thì, **Decoding** ở đây là cho trước một *HMM* $\lambda = (A, B)$ và một chuỗi quan sát $O = o_1, o_2, ..., o_T$; Nhiệm vụ ở đây là tìm

chuỗi trạng thái có thể xảy ra nhất $Q = q_1 q_2 ... q_T$.

Để làm được điều này và giảm thiểu chi phí tính toán (Thay vì phải đi xét tất cả các trước hợp của chuỗi) thì ở đây ta áp dụng thuật toán *Viterbi*. Dưới đây là hình minh họa đơn giản của thuật toán trên cho ví dụ về *Ice Cream*



Ở đây ta thấy *Viterbi* là một thuật toán dạng diễn tiến (*Forward*). Ta sẽ cùng đi tìm hiểu cụ thể về thuật toán này. Đầu tiên là các kí hiệu trong hình, π là phân phối xác suất đầu vào của tập các trạng thái ẩn với tổng các xác suất là 1. Các ô tròn chứa các trạng thái ẩn (H và C), và các ô vuông chứa các trạng thái quan sát (1, 2 hoặc 3) qua từng thời điểm t.

Đây là quá trình tính toán các giá trị $v_t(j)$ cho các trạng thái ẩn, giá trị này được hiểu là xác suất xuất hiện của các trạng thái tại thời điểm đó, dựa vào trạng thái ẩn (Q) ở thời điểm ngay trước kết hợp với trạng thái (O) quan sát được tại đó. Và sau đây là cách tính toán các giá trị trên.

Ở thời điểm đầu tiên, tương ứng với $t = 1$, các giá trị $v_t(j)$ của các trạng thái ẩn tương ứng được tính bằng tích xác suất đầu vào π_j của trạng thái ẩn đó nhân với xác suất có được trạng thái quan sát o_t từ trạng thái ẩn q_j đó. Ví dụ như trên hình ta thấy:

$$v_1(1) = P(C|start) * P(3|C) = .2 * .1 = .02$$

Từ thời điểm thứ 2 trở đi, tương ứng với $t \geq 2$, các $v_t(j)$ được tính tương tự như trên tạo thành các đường dẫn mũi tên đến các nút trạng thái ẩn.

Nhưng tại các nút trạng thái ẩn từ thời điểm thứ 2 trở đi sẽ có nhiều hơn 1 (cụ thể ở ví dụ này là 2) mũi tên trỏ vào, gọi là các đường dẫn. Cụ thể giá trị của các đường dẫn sẽ được tính theo công thức chung là:

$$v_{t-1}(i)a_{ij}b_j(o_t), \forall 1 \leq i \leq T$$

Với:

- T: Số lượng các trạng thái ẩn (Ở ví dụ này là 2)
- $v_{t-1}(i)$: Xác suất của trạng thái ẩn thứ i tại thời điểm thứ $t-1$
- a_{ij}: Xác suất chuyển tiếp từ trạng thái ẩn thứ i sang trạng thái ẩn thứ j
- $b_j(o_t)$: Xác suất có được trạng thái quan sát o_t khi biết trạng thái ẩn thứ j

Thông thường ta sẽ tính tổng xác suất các đường dẫn vào một nút để có được xác suất xuất hiện trạng thái ẩn đó. Nhưng đối với thuật toán *Viterbi* chúng ta sẽ không làm như vậy, vì mục đích của thuật toán này là tìm ra chuỗi trạng thái ẩn tốt nhất tương ứng với chuỗi trạng thái quan sát. Do đó việc cần làm ở đây là ta sẽ đi xác định và chọn đường dẫn nào có xác suất cao nhất làm giá trị tại nút hiện tại. Tức là:

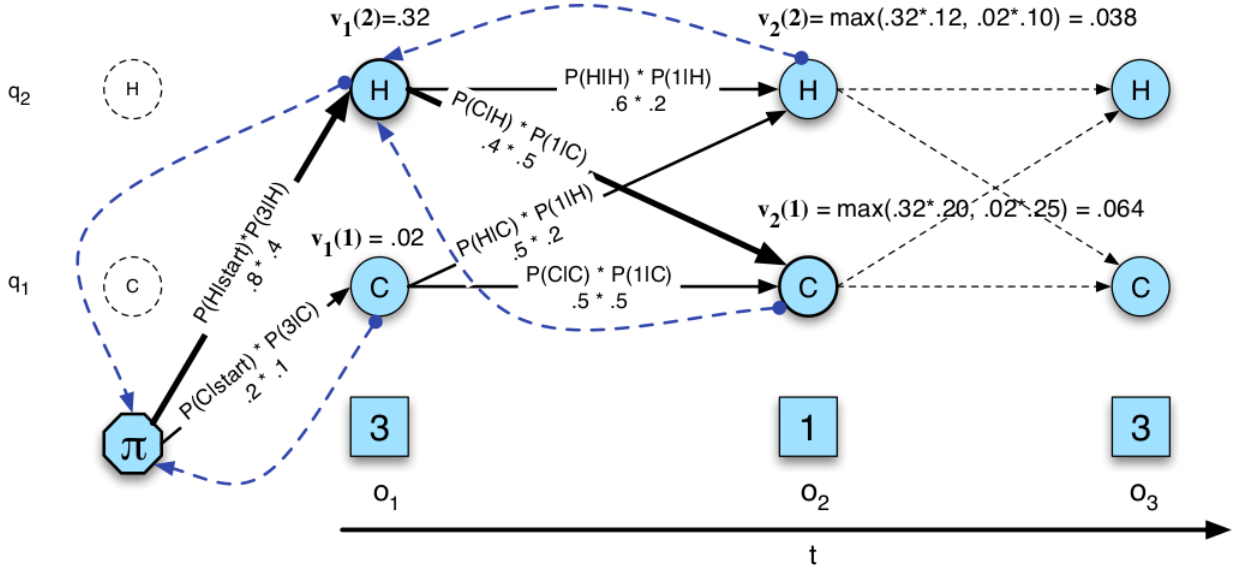
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)a_{ij}b_j(o_t)$$

Với ý tưởng như trên, thuật toán *Viterbi* được triển khai như sau:

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Trong đoạn mã giả trên ta thấy có một con trỏ **backpointers**. Mục đích của nó là lưu lại đường dẫn tốt nhất tới nút hiện tại. Tất cả được thể hiện trong hình minh họa dưới đây



3.2 HMM Training (Learning): Thuật toán Forward-Backward

Tiếp theo là vấn đề thứ 3 được nhắc tới ở trên: Học các tham số của *HMM* (Các ma trận A và B), từ việc cho trước chuỗi quan sát O và tập các trạng thái có thể có.

Đối với một mô hình HMM trong thực tế, thuật toán này sẽ bắt đầu bằng một ước tính cho các xác suất chuyển đổi và quan sát (A và B), sau đó sử dụng các xác suất ước tính này để thu được các xác suất ngày càng tốt hơn. Ta có thể thực hiện điều này bằng cách tính toán xác suất thuận (*Forward Probability*) cho một quan sát và sau đó chia đại lượng xác suất đó cho tất cả các đường dẫn khác nhau góp phần vào xác suất này.

Cụ thể hơn ở đây, ta có một định nghĩa mới là **Forward Probability** (tạm dịch là xác suất quay lui) β, là xác suất đạt được các trạng thái từ thời điểm $t+1$ cho tới cuối, khi ta đang ở trạng thái i tại thời điểm t (Cho trước λ):

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T | q_t = i, \lambda)$$

Nó được tính toán tương tự như thuật toán diễn tiến (*Forward Algorithm*). Bao gồm các bước:

- Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

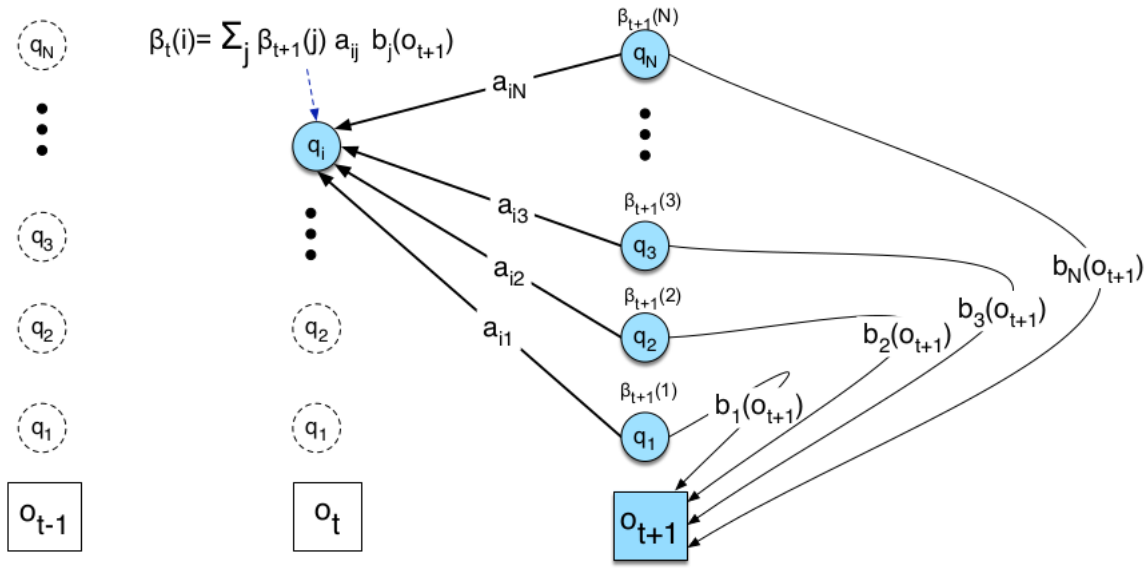
- Recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, \ 1 \leq t < T$$

- Termination:

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$$

Hình sau minh họa các bước của thuật toán quay lui nói trên:



Bây giờ chúng ta sẽ tìm hiểu cách để vận dụng xác suất diễn tiến và quay lui (*Forward and Backward*) để học mô hình, tức là xác định xác suất chuyển trạng thái ẩn a_{ij} (tương ứng với ma trận chuyển A) và xác suất của trạng thái quan sát $b_i(o_t)$

Đầu tiên, ta sẽ ước tính

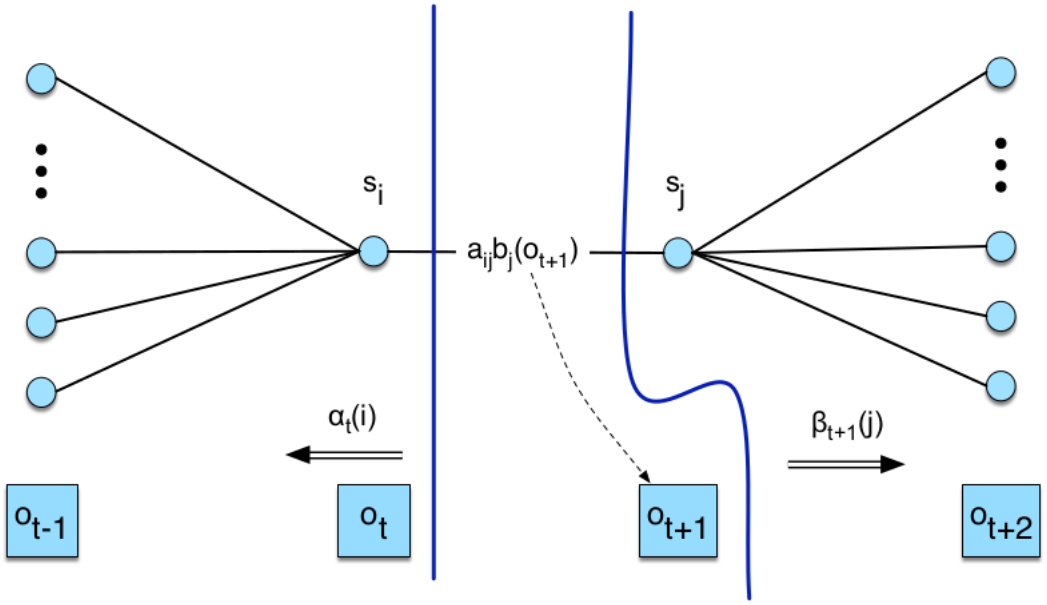
$$\hat{a}_{ij} = \frac{\text{Số lần chuyển đổi dự kiến từ trạng thái i sang trạng thái j}}{\text{Tổng số lần chuyển đổi dự kiến từ trạng thái i}}$$

Để tính được giá trị trên, trước hết ta định nghĩa $\xi_t(i,j)$ là xác suất để đạt trạng thái i tại thời điểm t và trạng thái j tại thời điểm $t+1$, cho trước mô hình và chuỗi quan sát:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O, \lambda)$$

Để tính được xác suất này, ta sẽ định nghĩa và tính một xác suất gần như tương tự nó, nhưng là bao gồm cả xác suất của trạng thái quan sát:

$$not - quite - \xi_t(i,j) = P(q_t = i, q_{t+1} = j, O|\lambda)$$



Nhìn vào hình minh họa trên, ta dễ dàng tính được:

$$not - quite - \xi_t(i,j) = \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$$

Để tính được ξ_t từ $not - quite - \xi_t$ ta dựa vào tính chất sau của xác suất:

$$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$$

Với $P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$. Khi đó ta tìm được:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

Khi đó *Số lần chuyển đổi dự kiến từ trạng thái i sang trạng thái j* chính là tổng tất cả các ξ tại tất cả thời điểm t. Và *Tổng số lần chuyển đổi dự kiến*

từ trạng thái i là tổng tất cả các lần chuyển đổi đi ra từ trạng thái i:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

Việc còn lại là đi tính ước lượng $\hat{b}_j(v_k)$, ở đây cho trước v_k từ tập hợp các quan sát V, và xét một trạng thái ẩn j:

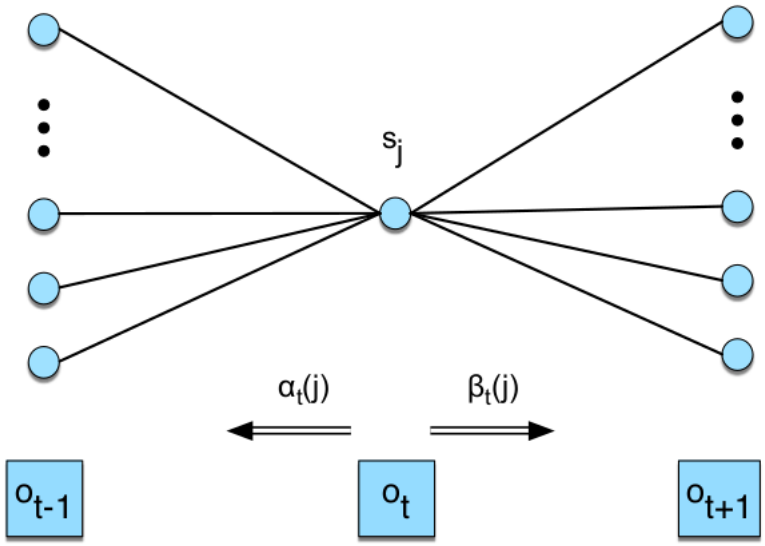
$$\hat{b}_j(v_k) = \frac{\text{Số lần dự kiến đang ở trạng thái j và có quan sát } v_k}{\text{Số lần dự kiến ở trạng thái j}}$$

Ở đây ta cần tính xác suất đạt trạng thái j tại thời điểm t, kí hiệu là:

$$\gamma_t(j) = P(q_t = j|O, \lambda)$$

Và một lần nữa tương tự như cách tính ở trên, ta được:

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)}$$



Dựa vào hình trên ta dễ dàng thấy $_t(j)$ tính thành:

$$\gamma_t(i) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

Bước cuối cùng là ta đi tính b:

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. o_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Với:

- Tử số là tổng $\gamma_t(j)$ qua tất cả các thời điểm t với quan sát o_t chính là ký tự v_k.
- Mẫu số là tổng $\gamma_t(j)$ qua tất cả các thời điểm t.

Kết quả thu được là tỷ lệ giữa số lần mà chúng ta đang ở trạng thái j và quan sát thấy v_k.
Vậy ta có thể thấy từ một chuỗi quan sát O cho trước và giả sử có được một ước tính trước của A và B, ta có thể dựa vào các phương trình sau để ước tính lại cho chính xác các giá trị A và B:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}, \quad \hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Dựa vào ý tưởng trên ta xây dựng được thuật toán *Forward-Backward* hoàn chỉnh, khởi đầu với việc khởi tạo một ước lượng cho HMM với tham số $\lambda = (A, B)$. Sau đó ta thực việc việc lặp đi lặp lại các bước tính toán: **expectation (E-step)** và **maximiztion (M-step)**. Hay có thể viết gọn lại là **EM**.

Ở **E-step**, ta tính toán các đại lượng γ và ϵ dự kiến từ các xác suất A và B được cho trước đó. Và tại **M-step**, ta lại dùng các đại lượng γ và ϵ vừa tìm được để đi tính lại các xác suất A và B mới. Ta xem xét mã giả của thuật toán như sau:

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

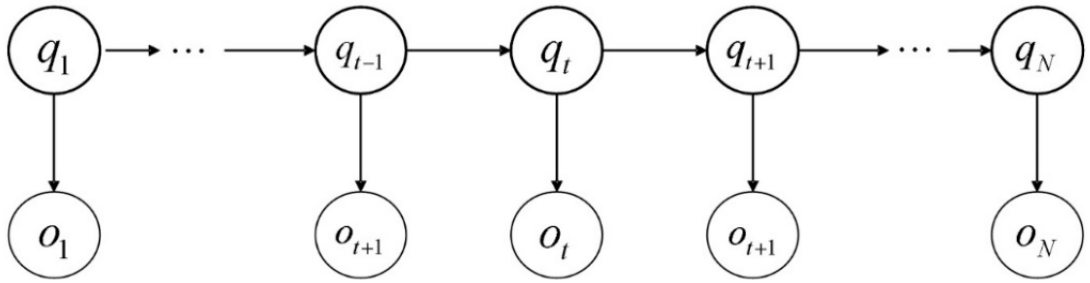
$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

3.3 Ứng dụng thuật toán HMM vào bài toán phân loại văn bản tiếng việt

Với dữ liệu văn bản chứa rất nhiều ký tự, đầu tiên ta phải tiền xử lý dữ liệu, loại bỏ những từ không cần thiết (*stop word*, những từ hiếm gặp, hay các dấu câu) để mô hình được xây dựng một các hiệu quả nhất. Công việc này sẽ được mô tả kĩ lưỡng hơn ở phần tiếp theo khi xây dựng mô hình *HMM* hoàn chỉnh. Sau quá trình này, tất cả dữ liệu của ta đều chỉ chứa những từ có nghĩa và quan trọng trong việc phân lớp.
Giả sử đoạn văn bản có độ dài là N từ. Ta sẽ biểu diễn lại đoạn văn bản dưới dạng tập hợp $O = \{o_1, o_2, ..., o_N\}$. O sẽ là một *Obsevation Sequence* trong *HMM*. Trong bài toán phân loại chủ đề văn bản này, ta sẽ cho *Hidden State* tương ứng của một *Observation* (tức là một từ) cũng chính là chủ đề của đoạn văn bản đang xét. Ứng với một *Obsevation Sequence* ta sẽ có một *Hidden State Sequence* $Q = \{q_1, q_2, q_3, ..., q_N\}$ với mọi q_i đều sẽ là chủ đề của đoạn văn bản đó.



Ví dụ, ta có một đoạn văn bản ngắn "Đội tuyển Việt Nam" thuộc phân loại *the-thao* (Thể thao). Thì ta sẽ có hai chuỗi tương ứng là $O = \{$'Đội', 'tuyển', 'Việt', 'Nam'$\}$ và $Q = \{$"thể thao", "thể thao", "thể thao", "thể thao"$\}$
Các bước tiếp theo ta sẽ vận dụng các thuật toán của *Hidden Markov Model* đã nêu ở trên vào bài toán này.

4 Hidden Markov Model hoàn chỉnh cho bài toán phân loại văn bản Tiếng Việt

4.1 Ngôn ngữ và các thư viện được dùng để cài đặt

Ngôn ngữ dùng để cài đặt ở đây là *Python* 3, với các thư viện hỗ trợ là:

- numpy
- pandas
- requests
- BeautifulSoup
- logging
- os
- time
- math
- matplotlib
- seaborn
- ViTokenizer
- Những thư viện trong sklearn:
  - TfidfVectorizer
  - train_test_split
  - LabelEncoder
  - accuracy_score
  - Pipeline
  - MultinomialNB
  - BaseEstimator, TransformerMixin, ClassifierMixin

4.2 Thu thập, khám phá và tiền xử lý dữ liệu

4.2.1 Thu thập dữ liệu

Như đã đề cập ở phần giới thiệu, ta sẽ tiến hành thu thập dữ liệu từ trang báo điện tử **vietnamnet.vn** để làm tập dữ liệu *train* và **baomoi.com** để làm tập dữ liệu *test*. Ta có thể thấy đây là hai trang báo lớn, đảm bảo được sự chính xác của dữ liệu. Kèm theo việc sử dụng dữ liệu của hai trang khác nhau làm hai tập *train* và *test* giúp đảm bảo được sự khách quan, đánh giá tốt hơn sự hiệu quả của mô hình, tránh tình trạng overfit.
Quá trình thu thập dữ liệu được thực hiện trong file **collect_data.ipynb**. Sau khi thu thập, dữ liệu dùng để huấn luyện được lưu vào file **data.txt** và dữ liệu test được lưu vào file **test.txt**. Dữ liệu lưu trữ sẽ là dữ liệu gắn nhãn, tương ứng với các chủ đề văn bản.

	Nội dung văn bản	Chủ đề
0	Bắn thuốc mê di dời đàn khỉ trong khu dân cư...	thoi-su
1	Lớp học bằng ván gỗ cũ nát ở vùng cao Lạng S...	thoi-su
2		thoi-su
3	Sài Gòn xuống 19 độ C Ảnh hưởng không khí lạ...	thoi-su
4		thoi-su
...	...	...
1832	Lễ hội áo dài sắp diễn ra bên bờ sông Hương ...	du-lich
1833	Trên máy bay không phải chỗ nào cũng cho trẻ...	du-lich
1834	Cắm trại đón Giáng sinh trong sân bay Singap...	du-lich
1835	Saigontourist Group khuyến mại lớn dịp Giáng...	du-lich
1836	Lợi ích khi là khách hàng thân thiết của côn...	du-lich

1837 rows × 2 columns

Tiếp tục chúng ta dùng phương pháp gom từ **n-gram** để được những từ ghép liên quan, có ý nghĩa hơn.

	Nội dung văn bản	Chủ đề
0	Bắn thuốc_mê di_dời đàn khỉ trong khu dân_cư S...	thoi-su
1	Lớp_học bằng ván gỗ cũ nát ở vùng_cao Lạng Sơn...	thoi-su
2		thoi-su
3	Sài_Gòn xuống 19 độ C Ảnh_hưởng không_khí lạnh...	thoi-su
4		thoi-su
...	...	...
1832	Lễ_hội áo_dài sắp diễn ra bên bờ sông Hương Th...	du-lich
1833	Trên máy_bay không phải chỗ nào cũng cho trẻ_e...	du-lich
1834	Cắm trại đón Giáng_sinh trong sân_bay Singapor...	du-lich
1835	Saigontourist_Group khuyến_mại lớn dịp Giáng_s...	du-lich
1836	Lợi_ích khi là khách_hàng thân_thiết của công_...	du-lich

4.2.2 Khám phá và tiền xử lý dữ liệu

Trong quá trình khám phá dữ liệu ta sẽ xem liệu các dòng dữ liệu có bị trùng nhau không, có dòng dữ liệu nào bị thiếu (tức là không lấy được văn bản từ *web* hay không. Và cuối cùng là liệu tập dữ liệu chúng ta sử dụng có đồng đều giữa các lớp (*class*) hay không, như chúng ta biết thì nếu một tập dữ liệu bị lệch thì sẽ dẫn đến mô hình học không hiệu quả.

⇒ Dữ liệu phân bố khá đều. Ta sẽ tiến hành bước tiếp theo, tiền xử lý dữ liệu.

Ở bước tiền xử lý dữ liệu, ta sẽ thực hiện hai công việc:

- Đầu tiên là xóa đi những dòng dữ liệu có quá ít kí tự (cụ thể ở đây là dưới 100). Những dòng này có thể là dữ liệu không lấy được vì không giống định dạng mẫu của hầu hết trang web nên code không lấy được dữ liệu hoàn chỉnh, cũng có thể là bài báo này chứa video là chính.

- Tiếp theo ta sẽ chuyển các chủ đề (Thuộc tính *class*) thành dạng số để thuận tiện cho việc phân lớp dữ liệu nhằm huấn luyện mô hình.

4.3 Xây dựng Hidden Markov Model

Để chuẩn bị cho việc xây dựng mô hình:

- Trước tiên ta phải thực hiện tách dữ liệu *train* thành hai tập là *X_ train* và *y_ train*.

- Tiếp theo là xác định các *hidden states* chính là tập các giá trị phân biệt của tập *y_ train*, ở trong mô hình của chúng ta thì sẽ có tập các *hidden states* là [0, 1, 2, 3, 4, 5].

- Sau đó là tập các quan sát (*Observations*) chính là tập tất cả các từ đơn hoặc ghép trong toàn bộ dữ liệu mà ta thu tập được dùng để huấn luyện.

	IDF
mâynhững	7.603944
stock	7.603944
liênbệnh	7.603944
liên_ấp	7.603944
liên_đội_trưởng	7.603944
...	...
là	1.088491
cho	1.083323
của	1.055725
trong	1.050723
và	1.012956

31230 rows × 1 columns

Tuy nhiên ở đây ta có thể thấy số lượng dữ liệu của tập quan sát quá nhiều (hơn 30,000 dòng), điều này sẽ dẫn tới việc hao tổn thời gian và bộ nhớ trong quá trình mô hình hóa dữ liệu. Chính vì vậy ta sẽ dựa vào lý thuyết TF-IDF để xác định và bỏ những *stopword* (những từ phổ biến, không quá quan trọng, thường xuyên xuất hiện trong tất cả các loại văn bản) cũng như những từ quá đặc biệt (Xuất hiện rất ít, chỉ vài lần trong tất cả văn bản của tập dữ liệu). Nhờ vào đó ta đã giảm được số lượng lớn cho tập quan sát.

	IDF
chọn_lựa	6.099866
nhảy_múa	6.099866
bình_tân	6.099866
bình_thạnh	6.099866
như_ý	6.099866
...	...
đầu	1.829392
cao	1.821658
gần	1.818581
2020	1.817046
lần	1.815514

5031 rows × 1 columns

- Và cuối cùng, ở mỗi dữ liệu, ta chỉ lọc và giữ lại những từ có trong tập quan sát (*Observations*).

```
0       [di_dời, đàn, khu, dân_cư, chức_năng, dự_kiến,...
1       [bằng, gỗ, cũ, nát, vùng_cao, trò, ngồi, học, ...
3       [xuống, 19, độ, không_khí, lạnh, phía, nhiệt_đ...
5       [lão, 72, tuổi, đại_học, trở_thành, ngành, hệ,...
6       [000, gia_súc, chết, rét, con, dân, tỉnh, chết...
                              ...
1829    [công_nhận, văn_hóa, bán, hàng, rong, di_sản, ...
1831    [đẹp, đường, giáp, biển, bảo_tồn, thiên_nhiên,...
1832    [áo_dài, sắp, diễn, bên, bờ, sông, hội, áo_dài...
1835    [khuyến_mại, lớn, dịp, dành, ưu_đãi, 35, khách...
1836    [khách_hàng, thân_thiết, công_ty, du_lịch, 18,...
Name: Nội dung văn bản, Length: 1475, dtype: object
```

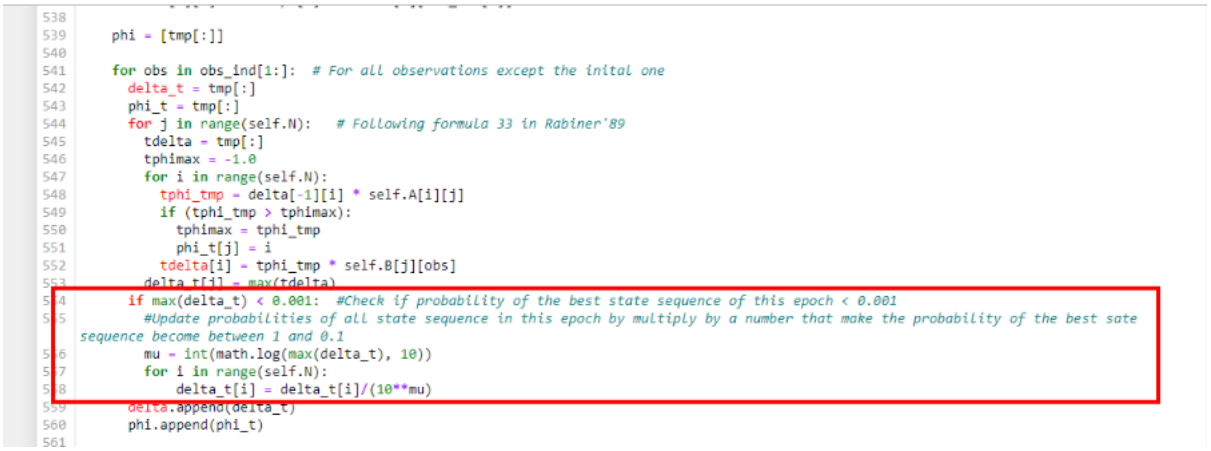
Sau khi thực hiện xong các bước chuẩn bị, ta kết hợp *X_ train (Observation Sequence)* và *y_ train (Hidden State Sequence)* để huấn luyện mô hình. Tới bước này, ta sẽ khởi chạy file **simplehmm.py** để có thể gọi hàm **hmm** đã cài đặt, nhằm huấn luyện mô hình Hidden Markov. Sau cùng ta được mô hình *HMM* hoàn chỉnh $\lambda = (\pi, A, B)$.
Lưu ý để phù hợp với bài toán phân loại văn bản này, ở đây ta sẽ thực hiện một số chỉnh sửa cho thuật toán **Viterbi**, cụ thể như sau:

- Thêm vào ở đầu hàm **Viterbi** (Thuật toán đã đề cập ở trên) đoạn *code* sau:

```
def viterbi(self, obser_seq):
  """Apply the Viterbi algorithm.

  USAGE:
    [sequence, seq_prob] = myhmm.viterbi(obser_seq)

  ARGUMENTS:
    obser_seq  An observation sequence (all observations must be in the list
               of observations of the HMM)

  DESCRIPTION:
    This routine uses the Viterbi algorithm to find the most likely state
    sequence for the given observation sequence. Returns the sequence in a
    list and it's probability.
  """

  for obs in obser_seq:
      if obs not in self.O:  #Check if word not in observation state
          #Add new word to observation_state
          self.O.append(obs)
          self.O_ind[obs] = self.M


          for i in range(self.N):  # For each state
              #Recalculate probabilities of each word in observation state
              for j in range(self.M):
                  self.B[i][j] = self.B[i][j]*self.M/(self.M + 1)
              #Add probability of new word to observation probabilities
              self.B[i].append(1/(self.M + 1))

          #Update observation state len
          self.M = self.M + 1
```

Mục đích: Hàm Viterbi không thể xử lý được trường hợp có từ mới không có sẵn trong *Observation States* nhưng trong thực tế thì trong một *Observation Sequence* bất kỳ có thể có các từ mà *Observation State* không có.
=> Thêm vào đoạn code xác suất của từ mới vào ma trận *Observation Probabilities*. Ở đây xác suất của từ mới sẽ được tính thông qua chuẩn hóa *Laplace* tức là 1/(số từ của *Observation State*) và do có thêm từ mới vào *Observation State* nên cần phải cập nhật lại xác suất của các từ cũ.

- Thêm vào trong vòng lặp tính các xác suất của các *State Sequence* đoạn code sau:



```
538
539     phi = [tmp[:]]
540
541     for obs in obs_ind[1:]:  # For all observations except the inital one
542       delta_t = tmp[:]
543       phi_t = tmp[:]
544       for j in range(self.N):   # Following formula 33 in Rabiner'89
545         tdelta = tmp[:]
546         tphimax = -1.0
547         for i in range(self.N):
548           tphi_tmp = delta[-1][i] * self.A[i][j]
549           if (tphi_tmp > tphimax):
550             tphimax = tphi_tmp
551             phi_t[j] = i
552           tdelta[i] = tphi_tmp * self.B[j][obs]
553         delta_t[j] = max(tdelta)
554       if max(delta_t) < 0.001:  #Check if probability of the best state sequence of this epoch < 0.001
555         #Update probabilities of all state sequence in this epoch by multiply by a number that make the probability of the best sate
    sequence become between 1 and 0.1
556         mu = int(math.log(max(delta_t), 10))
557         for i in range(self.N):
558             delta_t[i] = delta_t[i]/(10**mu)
559       delta.append(delta_t)
560       phi.append(phi_t)
561
```

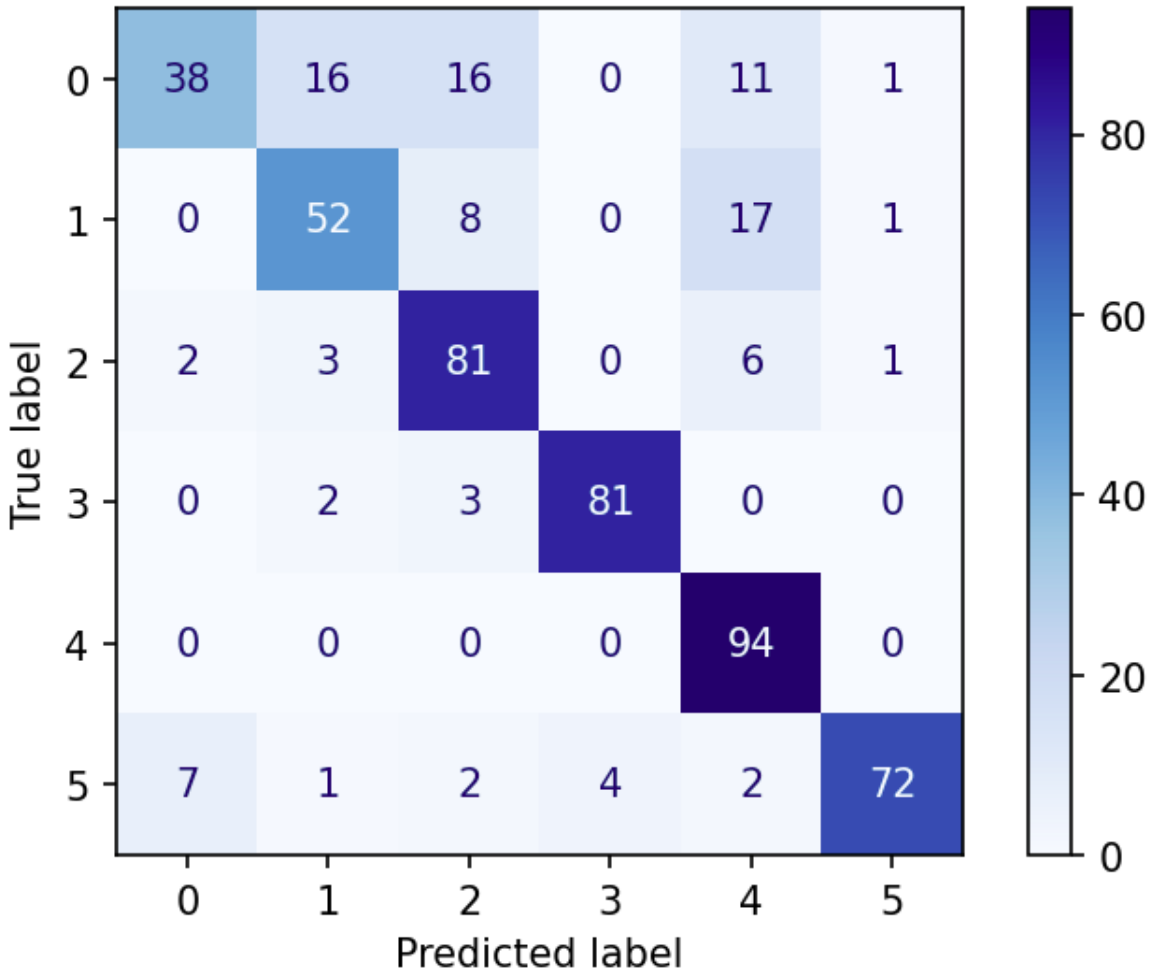
Mục đích: Trong lúc tính toán các xác suất của các *State Sequence* thì nếu số lượng từ quá lớn sẽ làm xác suất của các này rất nhỏ, quan trọng là điều này xảy ra với tất cả các *State Sequence* chứ không riêng một *State Sequence* đặc biệt nào => Sẽ có một *epoch* đủ lớn nào đó mà tất cả các *State Sequence* trong *epoch* đó đều bị làm tròn thành 0 => Ta sẽ không thể so sánh để tìm ra được *State Sequence* có xác suất lớn nhất.
Vì trong bài toán này ta không quan tâm đến xác suất của *State Sequence* tốt nhất tìm được => Có thể chuẩn hóa các xác suất của các *State Sequence* để không bị tình trạng quá nhỏ nữa. Ở đây nếu xác suất giảm xuống dưới một ngưỡng nào đó (trong *code* là 0.001) thì ta sẽ chuẩn hóa bằng cách nhân tất cả các xác suất của các *State Sequence* với một số để kết quả đạt được là *State Sequence* có xác suất lớn nhất trong *epoch* đó có giá trị ở giữa 1 và 0.1.

Tất cả các bước bao gồm tiền xử lý và huấn luyện mô hình ở trên, cũng như thử nghiệm mô hình ở bước tiếp theo trên đều được đặt trong một *Pipeline*.

4.4 Thử nghiệm và đánh giá hiệu suất của mô hình

4.4.1 Thử nghiệm mô hình với tập dữ liệu test

Như đã trình bày ở trên, dữ liệu *test* sẽ được lấy từ trang báo điện tử **baomoi.com**, khác với tập dữ liệu dùng để *train*. Đối với dữ liệu *test* ta cũng thực hiện các bước tiền xử lý tương tự như tập *train* để đảm bảo tính đồng bộ của dữ liệu. Rồi cuối cùng đưa tập dữ liệu *test* sau khi đã qua xử lý vào mô hình để dự đoán.



Ở đây ta thấy kết quả phân lớp khá chính xác. Với đường chéo chính là những ô đại diện cho việc phân lớp chính xác, có màu đậm hơn, tức là số lượng phân lớp chính xác nhiều hơn so với những trường hợp sai lệch. Độ chính xác ở đây được thể hiện qua hai thông số là:

- **Accuracy:** 0.8023032629558541
- **F1:** 0.7946336276050121

4.4.2 So sánh với thuật toán Naive Bayes Classification (NBC) và đánh giá hiệu suất của mô hình

a. Thuật toán Naive Bayes Classification (NBC) là gì?

Naive Bayes Classification (NBC) là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes, thuộc nhóm *Supervise Learning*. Thuật toán này dựa theo định lý *Bayes* quen thuộc:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Giả sử ta phân chia 1 sự kiện x thành n thành phần khác nhau $x_1, x_2, ..., x_n$. *Naive Bayes* theo đúng như tên gọi dựa vào một giả thiết **ngây thơ** rằng $x_1, x_2, ..., x_n$ là các thành phần độc lập với nhau. Từ đó ta có thể tính được:

$$P(x|y) = P(x_1 \cap x_2 \cap ... \cap x_n|y) = P(x_1|y)P(x_2|y)...P(x_n|y)$$

Do đó ta có được:

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

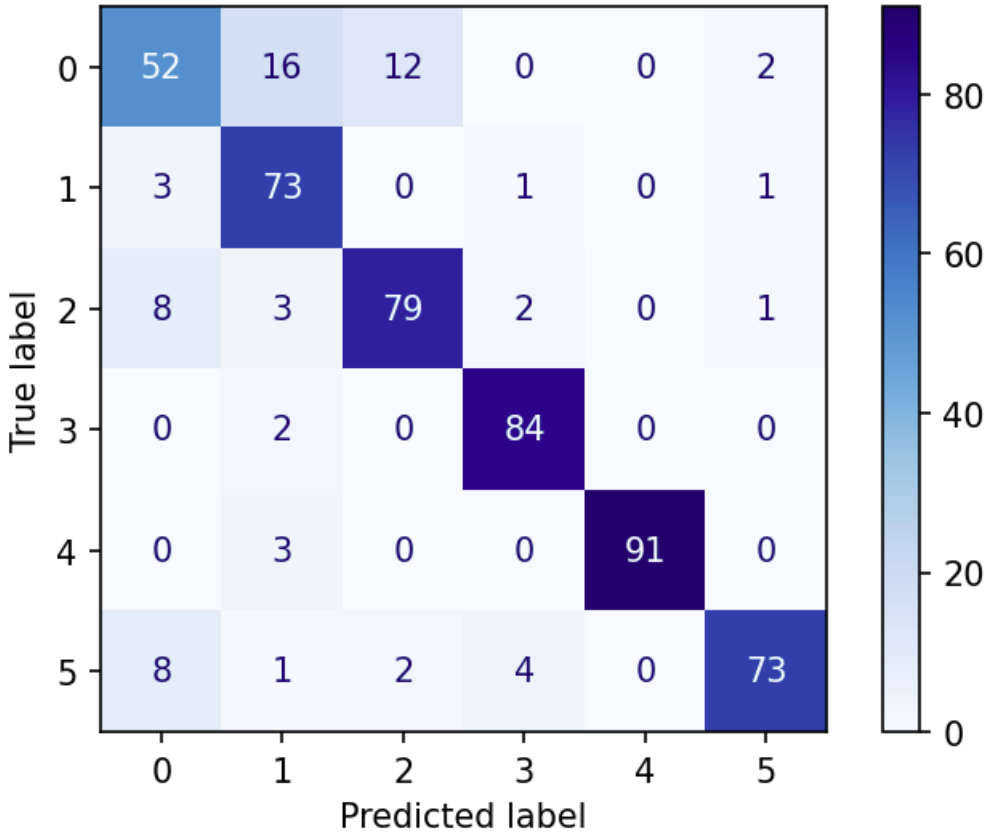
Với $\propto$ là phép tương quan thuận.

Trên thực tế thì ít khi tìm được dữ liệu mà các thành phần là hoàn toàn độc lập với nhau. Tuy nhiên giả thiết này giúp cách tính toán trở nên đơn giản, *training data* nhanh, đem lại hiệu quả bất ngờ với các lớp bài toán nhất định.

b. Áp dụng thuật toán NBC để phân lớp dữ liệu văn bản và so sánh kết quả với HMM đã xây dựng

Với mô hình thuật toán *NBC*, ta sử dụng hàm **MultinomialNB** có sẵn trong thư viện **sklearn.naive_bayes** rồi đưa vào *Pipeline* kèm với bước tiền xử lý dữ liệu trước đó.

Sau khi huấn luyện mô hình thành công, cũng như thử nghiệm lại mô hình với tập *test* ta được kết quả phân lớp với tập *test* như sau:



Nhìn qua đường chéo chính của hình trên, ta đã thấy mô hình này đạt được kết quả tương đối tốt hơn so với mô hinh *Hidden Markov* được xây dựng trong đồ án. Ta cùng đánh giá tổng quan độ chính xác của mô hình *NBC* vừa xây dựng qua 2 thông số là:

- **Accuracy:** 0.8675623800383877
- **F1:** 0.8664348724231457

c. Nhận xét và đánh giá các mô hình

Nhìn chung, mặc dù cùng dựa vào lý thuyết xác suất làm nền tảng để xây dựng mô hình, nhưng mô hình *Hidden Markov* cho kết quả phân lớp không tốt bằng mô hình *Naive Bayes Classification*. Không những vậy, *Hidden Markov Model* được xây dựng tương đối phức tạp, đòi hỏi chi phí thời gian và bộ nhớ lớn.

5 Tài liệu tham khảo.

- *Book: Speech and Language Processing - Daniel Jurafsky and James H. Martin*
- *Code tham khảo xây dựng Hidden Markov Model phân lớp dữ liệu văn bản*
- *Naive Bayes Classification (NBC) là gì?*