Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Análisis de datos complejos: Tarea 2

Jorge Luis Ramos Zavaleta

18 de septiembre de 2018

1. Ejercicio

Este ejercicio es sobre Hidden Markov Models (HMM).
Vimos en clase que el método más usado para el proceso de POS-tagging es HMM, donde asignamos etiquetas gramaticales POS (variables *latentes u ocultas*) a una secuencia de palabras (variables *observables*).
Dado un Corpus de entrenamiento y una secuencia de palabras de prueba, HMM calcula la *secuencia de etiquetas POS* más probable mediante la expresión

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}),$$

o, en palabras,

$$\hat{t}_1^n \approx \arg\max_{t_1^n} \prod P(\text{emisión})P(\text{transición}).$$

Estas probabilidades están totalmente definidas por el Corpus, y el proceso de entrenamiento de la red se realiza mediante operaciones de conteo de las palabras y las etiquetas contenidas.
Vamos a utilizar un *mini* Corpus tomado de la NYU[1], el cual puedes encontrar en el archivo `POSData.zip`, y contiene el corpus de entrenamiento (`training.pos`) y otro corpus de prueba (`development.pos`).

[1] `http://cs.nyu.edu/courses/spring12/CSCI-GA.2590-001/|`

1. Calcula las probabilidades de emisión y transmisión (verosimilitud y apriori) a partir del corpus de entrenamiento.

2. Considera el texto de prueba:

   ```
   Your contribution to Goodwill will mean more than you may know.
   ```

   Obten los tokens del texto de prueba. Verifica que, para el corpus de entrenamiento usado, las posibles etiquetas POS para cada token es:

   ```
   $Your
   [1] "PRP$"
   $contribution
   [1] "NN"
   $to
   [1] "IN" "TO"
   $Goodwill
   [1] "NNP"
   $will
   [1] "NN" "MD"
   $mean
   [1] "JJ"  "VB"  "VBP"
   $more
   [1] "RB"  "JJR" "RBR"
   $than
   [1] "IN"
   $you
   [1] "PRP"
   $may
   [1] "MD"
   $know
   [1] "NN"  "VB"  "VBP"
   $.
   [1] "COMMA" "."
   ```

3. Verifica que el número de posibles secuencias (*paths*) para nuestro sencillo texto de prueba es de 216. Esto te puede dar una idea de la complejidad computacional de éste tipo de problemas. Puedes verificarlo computacionalmente.

4. Usa el algoritmo Viterbi para estimar la secuencia de etiquetas POS del texto de prueba. Compara tu resultado con el obtenido al usar el Anotador de `coreNLP` de Stanford.

   En `R` puedes usar la función `viterbi(hmm,tokens)`, incluida en la librería `HMM`. Esta función recibe como parámetros un modelo HMM y el conjunto de tokens de prueba, y calcula la secuencia más probable de variabes ocultas, en nuestro caso, etiquetas POS.

   El modelo HMM debes definirlo con `initHMM()` usando las probabilidades apriori y verosimilitud del Corpus de prueba. Revisa la ayuda de la función.

5. Generalmente, no todas las palabras están incluidas en el Corpus de entrenamiento. Intenta, por ejemplo, asignar POS-tags al texto:

   `Coming to Goodwill was the first step toward my becoming totally`.

   ¿Qué podemos hacer en este caso? Implementa tu idea y verifica su desempeño con textos del corpus de prueba.

1.1. Solución

Para la primera parte del ejercicio se generaron las matrices de probabilidades emisión y de transmisión haciendo uso de los tags y de las palabras en el archivo training.pos. La matriz de probabilidades de transición se arma a partir de los etiquetados de las palabras, mientras que la matriz de probabilidades de emisión hace uso de ambas variables: palabras y tags para formarse.

Para la segunda parte del ejercicio se realizo un codigo que verifica cada palabra en el texto de prueba tokenizado y busca sus correspondientes etiquetas en el corpus de entrenamiento, con lo que obtuvieron los siguientes resultados:

```
[1] "Your tag->PRP$"
[1] "contribution tag->NN"
[1] "to tag->TO"
[1] "to tag->IN"
[1] "Goodwill tag->NNP"
[1] "will tag->MD"
[1] "will tag->NN"
[1] "mean tag->VBP"
[1] "mean tag->VB"
[1] "mean tag->JJ"
[1] "more tag->JJR"
[1] "more tag->RBR"
[1] "more tag->RB"
[1] "than tag->IN"
[1] "you tag->PRP"
[1] "may tag->MD"
[1] "know tag->VBP"
[1] "know tag->VB"
[1] "know tag->NN"
[1] ". tag->."
[1] ". tag->COMMA"
```

que coincide con los resultados esperados.

Para la tercera parte se podría hacer el calculo directamente al ver que las etiquetas coinciden con las planteadas, es un problema combinatorio muy simple, dado que la primera palabra (”Your”) solo tiene un tag, la segunda (çontribution”) tambien solo tiene un tag, la tercera (”to”) tiene 2, y así en general, se tiene que el número de secuencias posibles es entonces:

$$1 * 1 * 2 * 1 * 2 * 3 * 3 * 1 * 1 * 1 * 3 * 2 = 216$$

entonces se tienen 216 secuencias posibles.

Para la cuarta parte, usando CoreNLP se obtienen los siguientes resultados de taggeo sobre el texto de prueba:

```
> getToken(output)[,7]
 [1] "PRP$" "NN"   "TO"   "NNP"  "MD"   "VB"   "JJR"  "IN"   "PRP"  "MD"   "VB"   "."
```

Mientras que haciendo uso del algoritmo de Viterbi se obtienen los siguientes taggeos para el mismo texto de prueba:

```
> viterbi(hmm1,observation = token[[1]])
 [1] "PRP$" "NN"   "TO"   "NNP"  "MD"   "VB"   "JJR"  "IN"   "PRP"  "MD"   "VB"   "."
```

por lo que ambos esquemas de taggeo retornan las mismas etiquetas en este caso.

Para la última parte se considera que pueden haber en un texto palabras que no se encuentran en el corpus, por lo que usando cadenas de Markov ocultas como lo hemos venido haciendo implicara un error dado que la probabilidad de emisión sera igual a cero, por lo que debe encontrarse una forma de lidiar con este problema.

Una forma simple de lidiar con este problema es reasignar a todas las palabras que no se encuentren en el corpus de entrenamiento una palabra elegida e incluir en la matriz de probabilidades de emisión una columna asegurando una probabilidad distinto de cero a dicha palabra para que el algoritmo de Viterbi la pueda reconocer y etiquetarla como a las otras.

En nuestro caso las palabras desconocidas se sustituyeron por la palabra ÜKNçon lo que palabras que no estaban en el corpus ahora reciben una etiqueta. Dicha solución se probo primero en un texto de prueba
`Coming to Goodwill was the first step toward my becoming totally`.
y se obtuvo lo siguiente:

```
laplace("Coming to Goodwill was the first step toward my becoming totally."
        ,Prob_Emission2,prob_trans_tags)

             tokens_orig
 [1,] "NN"   "Coming"
 [2,] "TO"   "to"
 [3,] "NNP"  "Goodwill"
 [4,] "VBD"  "was"
 [5,] "DT"   "the"
 [6,] "JJ"   "first"
 [7,] "NN"   "step"
 [8,] "IN"   "toward"
 [9,] "PRP$" "my"
[10,] "VBG"  "becoming"
[11,] "NN"   "totally"
[12,] "."    "."
```

donde la palabra *totally* no se encuentra en el corpus, pero en este caso logra obtener una etiqueta.

Para terminar el ejercicio se probo la implementación en el archivo developer.pos con las palabras 51 a 100 ya que son las primeras en mostrar palabras desconocidas, con lo que se obtuvo

```
> laplace(paste0(devPos$word[51:100], collapse=" "),Prob_Emission2,prob_trans_tags)
 [1] "is"           "so"           "important"    "."            "Your"         "gift"
 "to"           "Goodwill"
 [9] "is"           "important"    "because"      "people"       "with"         "physical"
 "and"          "UKN"
[17] "disabilities" "sometimes"    "need"         "an"           "extra"        "hand"
"to"           "know"
[25] "the"          "pride"        "that"         "comes"        "with"         "work"
"."            "‘‘"
[33] "I"            "was"          "sad"          "when"         "I"            "could"
"n’t"          "go"
[41] "to"           "the"          "UKN"          "bar"          "to"           "buy"
"a"            "UKN"
[49] "."            "Now"
             tokens_orig
 [1,] "VBZ"  "is"
 [2,] "RB"   "so"
 [3,] "JJ"   "important"
```

```
 [4,] "."    "."
 [5,] "PRP$" "Your"
 [6,] "NN"   "gift"
 [7,] "TO"   "to"
 [8,] "NNP"  "Goodwill"
 [9,] "VBZ"  "is"
[10,] "JJ"   "important"
[11,] "IN"   "because"
[12,] "NNS"  "people"
[13,] "IN"   "with"
[14,] "JJ"   "physical"
[15,] "CC"   "and"
[16,] "JJ"   "Mental"
[17,] "NNS"  "disabilities"
[18,] "RB"   "sometimes"
[19,] "VB"   "need"
[20,] "DT"   "an"
[21,] "JJ"   "extra"
[22,] "NN"   "hand"
[23,] "TO"   "to"
[24,] "VB"   "know"
[25,] "DT"   "the"
[26,] "NN"   "pride"
[27,] "WDT"  "that"
[28,] "VBZ"  "comes"
[29,] "IN"   "with"
[30,] "NN"   "work"
[31,] "."    "."
[32,] "``"   "``"
[33,] "PRP"  "I"
[34,] "VBD"  "was"
[35,] "JJ"   "sad"
[36,] "WRB"  "when"
[37,] "PRP"  "I"
[38,] "MD"   "could"
[39,] "RB"   "n't"
[40,] "VB"   "go"
[41,] "TO"   "to"
[42,] "DT"   "the"
[43,] "JJ"   "snack"
[44,] "NN"   "bar"
[45,] "TO"   "to"
```

```
[46,] "VB"   "buy"
[47,] "DT"   "a"
[48,] "NN"   "soda"
[49,] "."    "."
[50,] "RB"   "Now"
```

En este caso las palabras 16, 43 y 48 no se encuentran en el corpus, dichas palabras corresponden a **mental** que la etiqueta como adjetivo, **snack** que la etiqueta igualmente como adjetivo, y la palabra **soda** que la etiqueta como pronombre, por lo que logra el cometido de etiquetar palabras que no se encuentran en el corpus.