STATISTICAL STRUCTURED PREDICTION
Question set (Part 1-A)

Question 1.

El algoritmo Outside calcula la posibilidad de que una cadena de símbolos $x_1 \ldots x_T$ sea generada a partir de un axioma S por una gramática incontextual, dado un no terminal *A* que genera una subcadena de longitud *l* que comienza en la posición x_i de la cadena completa. Para ello el algoritmo desarrolla todas las posibles derivaciones permitidas por una gramática incontextual que generen todos los símbolos de la cadena inicial que queden tanto a la izquierda del no terminal *A* como a su derecha, por lo tanto el caso base de este algoritmo será aquel en el que no exista ningún símbolo que derivar fuera de *A*. Por lo tanto la inicialización de este algoritmo será:

$$f(A, 0, T) = \delta(A, S)$$

Ya que para que se cumpla la condición de que no existan símbolos de la cadena fuera de *A*, esta debe ser de longitud igual a la totalidad de la cadena ( ]0,T]). Dado que el objetivo del algoritmo es obtener la probabilidad de que la cadena sea generada por la gramática a partir de *S*, si *A* es igual a *S*, el resultado de este algoritmo para una cadena A de longitud máxima debe ser 1 y 0 en otro caso.

Para el caso de la expresión que define el resultado total:

$$P_\theta(x) = \sum_{A \in N} f(A, i, i+1) \cdot p(A \to b) \cdot \delta(b, x_{i+1}) \quad 0 \leq i \leq T-1$$

Debemos calcular recursivamente el algoritmo Outside a partir de un caso en el que un no terminal *A* genere el símbolo *b* en la posición x_{i+1} de la cadena total, ya que podemos expresar fácilmente la probabilidad de que este símbolo sea generado mediante la expresión parcial:

$$p(A \to b) \cdot \delta(b, x_{i+1})$$

Y la probabilidad del resto de la cadena será generada por el algoritmo Outside. El resultado será el mismo independientemente de la *i* que se emplee y no es necesario realizar un sumatorio para todas los posibles valores de *i,* puesto que algunas de las probabilidades asociadas a un valor de *i* ya están contempladas en las probabilidades de otro valor de *i*.

Question 2.

Para obtener una versión del algoritmo Viterbi para Conditional Random Fields que devuelva la secuencia óptima asociada al mejor score debemos modificar los pasos de inicialización, recursión y resultado final del algoritmo de la siguiente forma:

Inicialización:

$$\forall s \in \mathrm{Y}$$

$$\gamma_1(s) = \psi_1(y_0 = null, y_1 = s, x_1)$$

$$s_0 = s$$

Recursión:

$$\forall s \in \mathrm{Y};\ \forall \mathrm{t} = 2 \ldots \mathrm{T}$$

$$\gamma_t(s) = \max_{s' \in \mathrm{Y}} \{\gamma_{t-1}(s) \cdot \psi_t(y_{t-1} = s', y_t = s, x_t)\}$$

$$s_{t-1} = \operatorname*{argmax}_{s' \in Y}(\gamma_t(s))$$

Resultado Final:

$$\max_s \gamma_T(s)$$

$$s_T = \operatorname*{argmax}_s \gamma_T(s)$$

Donde la secuencia óptima quedaría almacenada en la cadena $s_0 \ldots s_T$

Question 3.

Para obtener una versión del algoritmo Forward para HMM de trigramas deberemos aplicar los siguientes cambios:

Definición:

$$Score\ for\ x_1 .. x_t\ ending\ at\ y_t = s\ \in Y$$

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{y_1^t; y_t = s} \prod_{i=1}^{t} \psi_i(y_{i-2}, y_{i-1}, y_i, x_1)$$

Inicialización:

$$\forall s\ \in Y$$

$$\alpha_1(s) = \psi_1(y_{-1} = null, y_0 = null, y_1 = s, x_1)$$

$$\alpha_2(s) = \sum_{s' \in Y} \alpha_1(s') \cdot \psi_2(y_0 = null, y_1 = s', y_2 = s, x_2)$$

Recursión:

$$\forall t = 2..T \quad and \quad \forall s \in Y$$

$$\alpha_t(s) = \sum_{s\prime \in Y} \sum_{s\prime\prime \in Y} \alpha_{t-1}(s') \cdot \alpha_{t-2}(s'') \cdot \psi_t(y_{t-2} = s'', y_{t-1} = s', y_t = s, x_t)$$

La expresión del resultado final no necesita cambios:

$$\sum_s \alpha_T(s)$$

Question 4.

Nueva versión del algoritmo Inside para gramáticas *left-branching*:

La inicialización será igual a la del algoritmo original:

$$\forall A \in N; \ \ \forall i: 0 \ldots T-1;$$

$$e(A, i, i+1) = p(A \to b) \cdot \delta(b, x_{i+1})$$

La nueva expresión para la recursión será:

$$\forall A \in N; \ \ \forall l: 2 \ldots T; \ \ \forall i: 0 \ldots T-l;$$

$$e(A, i, i+l) = \sum_{B,C \in N} \{p(A \to BC) \cdot e(B, i, i+l-1).\, e(C, i+l-1, i+l)\}$$

Ya que el no terminal C siempre generará un símbolo terminal y por lo tanto solo tendrá en cuenta una cadena de longitud 1, desde *i+l-1* a *i+l.*

La expresión del resultado final sigue siendo la misma:

$$P_\theta(x) = e(S, 0, T)$$

Dado que con la nueva expresión de la recursión hemos eliminado un sumatorio para cada posible longitud de *B* y *C*, el coste del algoritmo pasa a ser:

$$O(n^2)$$

Practical Assigment

1. Statistical Evaluation

En este primer apartado de la tarea práctica se van a evaluar los diferentes modelos y submodelos ya entrenados en base a sus perplejidades y log-verosimilitud con respecto a 3 conjuntos de prueba diferentes.

1.1. Resultados empleando algoritmo Inside

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G1-EQ	TS-EQ	99095.20	-11503.83
G1-IS	TS-IS	26581.81	-10187.98
G1-SC	TS-SC	33618.35	-10422.82

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G2-EQ	TS-EQ	635136.42	-13361.59
G2-IS	TS-IS	52217.59	-10863.17
G2-SC	TS-SC	43543.30	-10681.51

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G3-EQ	TS-EQ	986.20	-6893.86
G3-IS	TS-IS	1254.89	-7134.80
G3-SC	TS-SC	1218.03	-7104.99

Tablas de perplejidad log-verosimilitud empleando el algoritmo Inside

La perplejidad puede interpretarse como un indicador de como de acertada es la predicción de un modelo clasificador con respecto a un conjunto de prueba, cuanto mayor es el valor de la perplejidad, peores predicciones realiza dicho modelo. Por otro lado, la log-verosimilitud indica cuan bien se un modelo se adapta a la distribución de un conjunto de observaciones, en este caso, al conjunto de test. Cuanto menor sea el valor de la log-verosimilitud peor será su adaptación al conjunto de test.

Si comparamos entre sí los resultados de los 3 modelos, podemos observar que los mejores resultados de perplejidad y log-verosimilitud se obtienen para el modelo G3 para todos sus submodelos. Esto indica, este modelo se encuentra mejor adaptado a los datos que han sido empleados como test y por lo tanto será capaz de distinguir y clasificar las muestras con mayor seguridad que el resto de modelos.

Otro factor interesante que podemos observar son los altos valores de perplejidad que se obtienen para el submodelo encargado de detectar triángulos equiláteros en los 2 primeros modelos con respecto al resto de sus respectivos submodelos.

1.2. Resultados empleando algoritmo Viterbi

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G1-EQ	TS-EQ	137806.35	-11833.60
G1-IS	TS-IS	35036.63	-10464.14
G1-SC	TS-SC	47517.29	-10768.84

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G2-EQ	TS-EQ	3575405.00	-15089.58
G2-IS	TS-IS	330794.74	-12709.25
G2-SC	TS-SC	311587.95	-12649.43

Modelo-Submodelo	Conjunto Test	Perplejidad	Log-Verosimilitud
G3-EQ	TS-EQ	12230868.83	-16319.47
G3-IS	TS-IS	47938195.82	-17685.42
G3-SC	TS-SC	45218066.24	-17627.00

Tablas de perplejidad log-verosimilitud empleando el algoritmo Viterbi

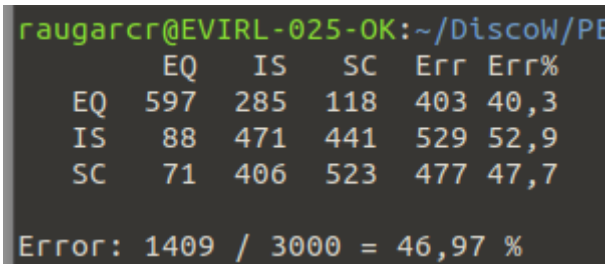
Si en lugar de emplear el algoritmo Inside empleamos el de Viterbi para calcular las probabilidades, vemos que todos los valores de perplejidad han aumentado con respecto a los resultados del algoritmo Inside. Esto se debe a que el algoritmo Viterbi solo devuelve la mayor probabilidad de todas las posibles secuencias de estados y no la suma de probabilidades, por lo que siempre será igual o menor a la probabilidad devuelta por el algoritmo inside. En especial parecen haber empeorado los resultados del modelo G3, que han pasado de ofrecer los mejores valores de perplejidad con Inside a los peores con Viterbi.

2. Clasification

En este apartado, guiándonos por las probabilidades empleadas en el apartado anterior para obtener la perplejidad y la log-verosimilitud, se ha construido una matriz de confusión en base a como clasifica cada modelo la totalidad de las muestras de test (3000 datos, 1000 por cada tipo de triángulo).

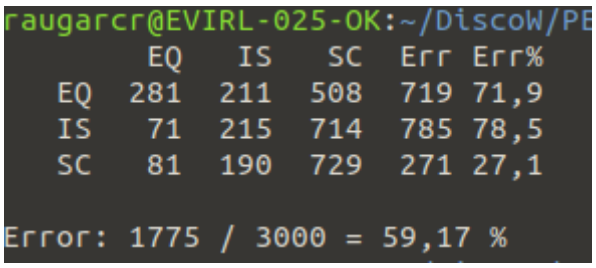
En estas matrices de confusión se representa en cada fila los resultados de clasificación para un conjunto de test (siguiendo el orden TS-EQ, TS-IS y TS-ES) junto al error y el porcentaje de error cometidos.

Dados los resultados del apartado anterior, se van a mostrar tan solo las matrices de confusión obtenidas a partir de las probabilidades del algoritmo Inside, ya que como ya se ha visto todos los valores de perplejidad empeoran al emplear Viterbi y lo mismo sucede con las matrices de confusión obtenidas a partir de estas probabilidades.



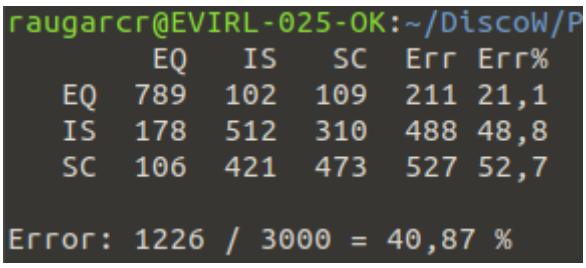
Matriz de confusión para el modelo G1

En la matriz de confusión del modelo G1 podemos observar que a pesar de que el peor valor de perplejidad para G1 se encontraba en el submodelo dedicado a detectar triángulos equiláteros, es para este tipo de triángulos para el que menos error se ha obtenido durante la clasificación. Adicionalmente podemos observar que al clasificar triángulos isósceles y escalenos, este modelo tiende a confundir ambos, ya que como se representa en la matriz de confusión clasifica erróneamente más de 400 muestras de triángulos isósceles como escaleno y viceversa, pero en comparación apenas clasifica estos como equiláteros.



Matriz de confusión para el modelo G2

Para el caso del modelo G2, la matriz de confusión muestra que este tiende a clasificar gran parte de las muestras como triángulos escalenos, independientemente de su clase real, lo que provoca gran porcentaje de error al clasificar triángulos equiláteros e isósceles, pero un bajo porcentaje al clasificar triángulos escalenos. Este bias al clasificar los triángulos provoca que en comparación con los resultados de G1, este modelo clasifique erróneamente un mayor porcentaje de las muestras de test, un 59 % frente al 47% obtenido para el modelo G1.



Matriz de confusión para el modelo G3

Tal y como se ha concluido en el apartado anterior en base a los resultados de la perplejidad, este modelo es el que mejor se adapta a los datos de test lo que se traduce en un menor porcentaje de error con respecto a los otros 2 modelos.

Aun así este modelo presenta el mismo problema que el modelo G1, confundiendo gran parte de los triángulos isósceles con escalenos y viceversa.