Encoder-Decoder Beam Search

Orlando Ramos Flores

Contenido

- Encoder-Decoder
  - Greedy Search
  - Search Tree
  - Beam Search

Greedy search

- Elegir el token único más probable para generar en cada paso se denomina decodificación ávida (greedy decoding); un algoritmo greedy es aquel que hace una elección que es localmente óptima, ya sea que resulte o no haber sido la mejor opción en retrospectiva.
- De hecho, una búsqueda ávida (greedy search) no es óptima y es posible que no se encuentre la traducción de mayor probabilidad.
- ¡El problema es que el token que ahora se ve bien para el decodificador podría resultar más tarde haber sido la elección incorrecta!

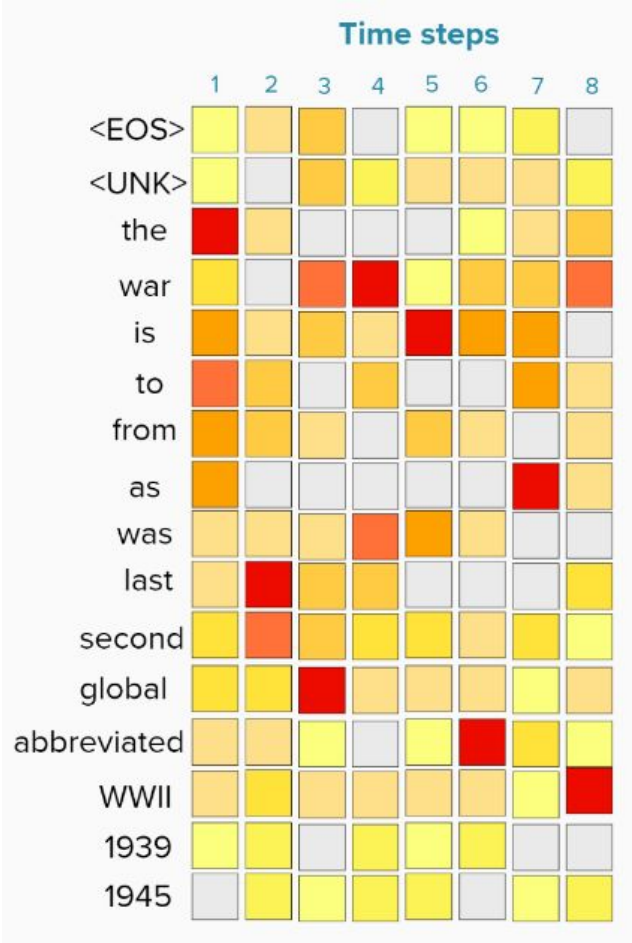
Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Greedy search

- La búsqueda ávida consiste en tomar el token con mayor probabilidad condicional del vocabulario **V**.

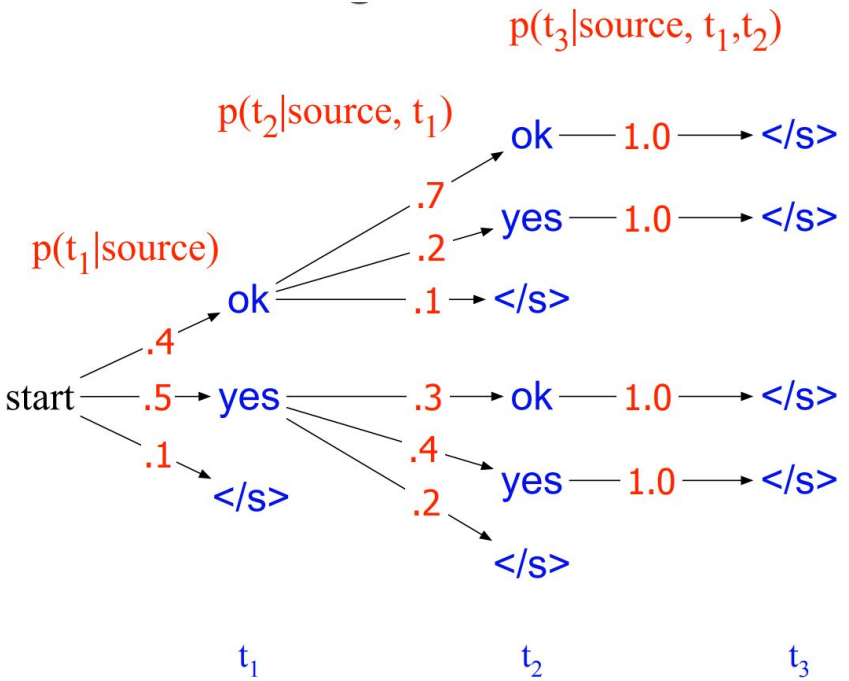
$$y_t = argmax_{y \in V} P(y|y_1, y_2, \ldots, y_{t-1}, x)$$

- En cada paso de tiempo $y_{(t)}$, el decodificador calcula la probabilidad de cada token en el vocabulario V.
- En consecuencia, cuanto más tokens haya en el vocabulario, más aumenta el costo computacional.
- Los cuadrados resaltados en rojo corresponden a las palabras que tienen la mayor probabilidad condicional en cada paso de tiempo t.
- Por lo tanto, la salida de secuencia predicha por el decodificador es: **the last global war is abbreviated as WWII**.
- Oculta altas probabilidades de que se puedan encontrar en tokens posteriores. Por lo tanto, no siempre genera secuencias de salida óptimas.



Search Tree

- Un árbol de búsqueda para generar la cadena de destino **T = t_1,t_2,...** a partir del vocabulario **V = {yes, ok, <s>}**, dada la cadena de origen.
- Muestra la probabilidad de generar cada token a partir de ese estado (nodos).
- La **greedy search** elegiría “yes” en el primer paso de tiempo seguido de “yes”, en lugar de la secuencia globalmente más probable “ok ok”.
- El único método garantizado para encontrar la mejor solución es una búsqueda exhaustiva: calcular la probabilidad de cada una de las oraciones posibles de V^T (para algún valor de longitud T) que obviamente es demasiado lento.



Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Ng, andrew. 2018. C5W3L03 Beam Search

Beam Search

- En Beam Search, en lugar de elegir el mejor token para generar en cada paso de tiempo, se mantiene **k** posibles tokens en cada paso.
- Esta huella de memoria de tamaño fijo **k** se denomina **ancho del beam** (*beam size*), en la metáfora del haz de luz de una linterna que se puede parametrizar para que sea más ancho o más estrecho.
- El beam size es el número de tokens con las probabilidades condicionales más altas en cada paso de tiempo **t**.

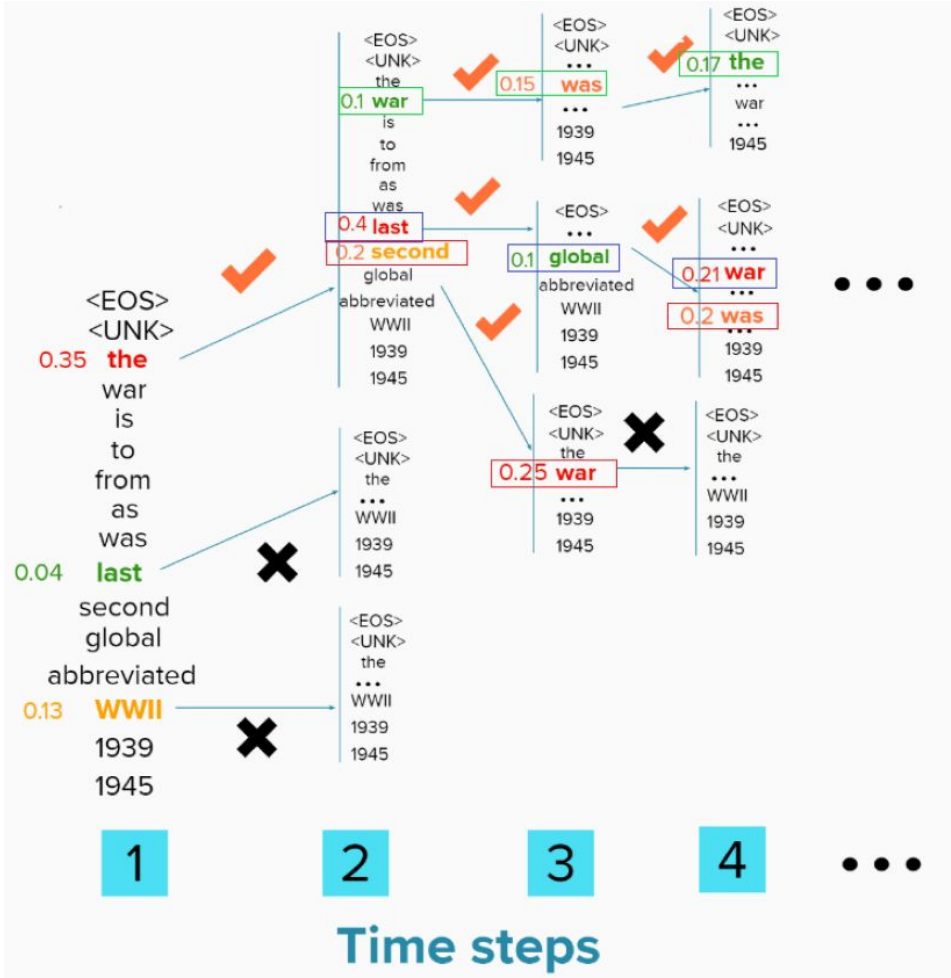
Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search: *beam size=3*

- **Sequence 1**: the last global war —
  (0.35 * 0.4 * 0.1 * 0.21) = 0.0029

- **Sequence 2** — the second war was —
  (0.35 * 0.2 * 0.25 * 0.2) = **0.0034**

- **Sequence 3** — the war was the —
  (0.35 * 0.1 * 0.15 * 0.17) = 0.00089



Beam Search

- Así, en el primer paso de la decodificación, se calcula un ***softmax*** sobre todo el vocabulario, asignando una probabilidad a cada palabra.
- Luego se seleccionan las **k-mejores** opciones de esta salida de ***softmax***.
- Estas **k salidas** iniciales son la frontera de búsqueda y estas **k palabras** iniciales se denominan **hipótesis**.
- Una **hipótesis** es una secuencia de salida (una traducción hasta ahora), junto con su probabilidad.

Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search

- En los pasos subsiguientes, cada una de las **k mejores hipótesis** se amplía de forma incremental al pasar a distintos decodificadores, cada uno de los cuales genera un **softmax** sobre todo el vocabulario para extender la hipótesis a todos los tokens siguientes posibles.
- Cada una de estas hipótesis **k∗V** se calcula mediante $\mathbf{P(y_i|x, y_{<i})}$: el producto de la probabilidad de la elección actual de palabras multiplicada por la probabilidad del camino que condujo a ella.
- Luego se podan las hipótesis **k∗V** hasta las **k mejores hipótesis**, de modo que nunca haya más de **k hipótesis** en la frontera de la búsqueda, y nunca más de **k decodificadores**.

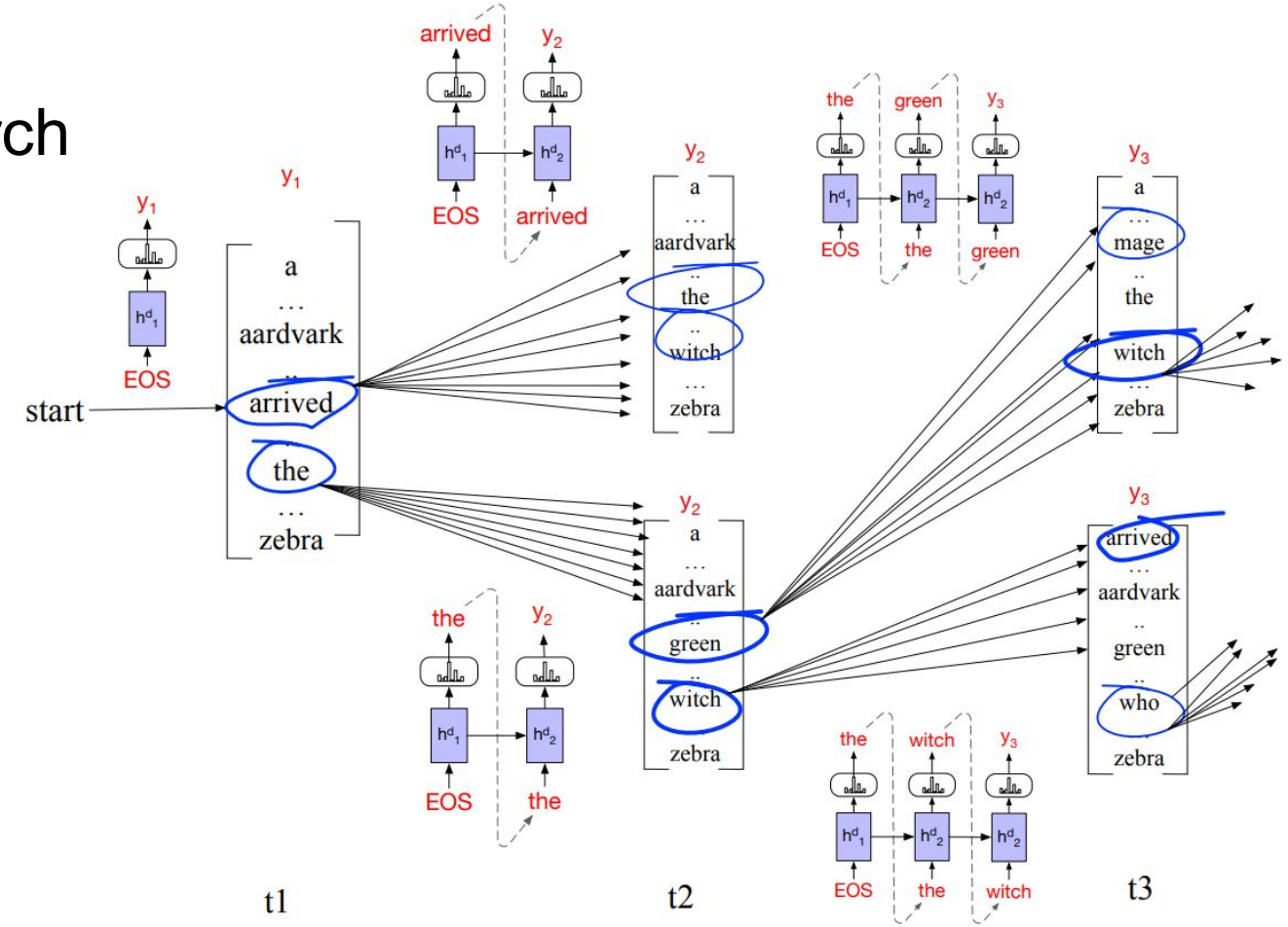
Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search

- Este proceso continúa hasta que se genera un token **</s>** que indica que se ha encontrado una salida candidata completa.
- En este punto, la hipótesis completada se elimina de la frontera y el ***beam size*** se reduce en uno. La búsqueda continúa hasta que el ***beam size*** se ha reducido a 0.
- El resultado serán **k hipótesis**.

Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search



beam size = 2

Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search

- ¿Que pasa si tenemos una secuencia de salida de longitud = 10,000?.
- Para tener la probabilidad condicional final multiplicamos $P(y_1) * P(y_2) * P(y_3) * \ldots * P(y_{10000})$, y el resultado será un número demasiado pequeño.
- Las computadoras se pueden limitar a representarlo, y la consecuencia es que perdemos precisión.
- Para abordar este problema, se usa la propiedad del producto de los logaritmos para convertir la multiplicación en una suma.

$$log(xy) = log(x) + log(y)$$

Ng, andrew. 2018. C5W3L03 Beam Search

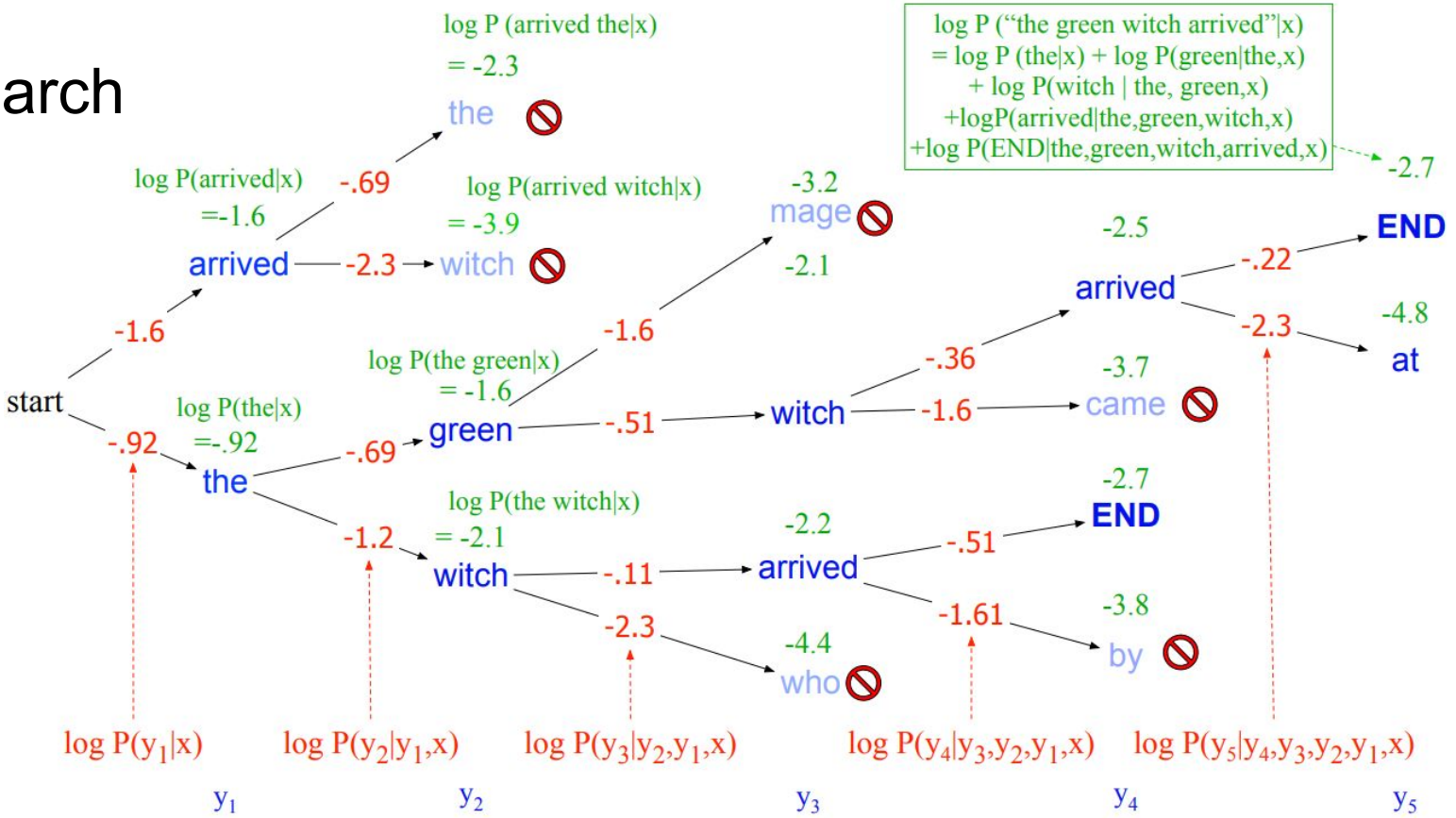
Beam Search

- Por lo tanto, la probabilidad de las secuencias de salida se calculan:

$$
\begin{aligned}
score(y) &= \log P(y|x) \\
&= \log\left(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)...P(y_t|y_1,...,y_{t-1},x)\right) \\
&= \sum_{i=1}^{t} \log P(y_i|y_1,...,y_{i-1},x)
\end{aligned}
$$

Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall.*
Ng, andrew. 2018. C5W3L03 Beam Search

Beam Search



Jurafsky, D., & Martin, J. H. (2021). Speech and language processing. Vol. 3. *US: Prentice Hall*.

Beam Search: Inconvenientes

- Al aumentar el ***beam size***, la calidad de la secuencia de salida mejora significativamente, pero reduce la velocidad del decodificador.
- Hay un punto de saturación, donde incluso si aumentamos el número del ***beam size***, la calidad de la decodificación ya no mejora.
- No garantiza encontrar la secuencia de salida con la puntuación más alta.

Ng, andrew. 2018. C5W3L03 Beam Search