Modelos de lenguaje basados en N-gramas

Rubén Francisco Manrique
rf.manrique@uniandes.edu.co

Modelos de lenguaje Probabilísticos

- El objetivo es asignar una probabilidad a una oración
  - Traducción:
    - P(Al que madruga Dios le ***ayuda***) > P(Al que madruga Dios le ***apoya***)
  - Corrección ortográfica:
    - NLP es un ***cursor*** de maestría
    - P(NLP es un ***curso*** de maestría) > P(NLP es un ***cursor*** de maestría)
  - Reconocimiento de voz:
    - P(Yo vi una vaca) > P(Llovía vaca)
  - Útil también en resúmenes automáticos, respuesta a preguntas, etc.

Modelos de lenguaje Probabilísticos

- Objetivo: calcular la probabilidad de una oración/sentencia dada como una secuencia de palabras:
  - P(W) = P(w1,w2,w3,w4,w5…wn)

- Tarea relacionada: probabilidad de una próxima palabra.
  - P(w5|w1,w2,w3,w4)

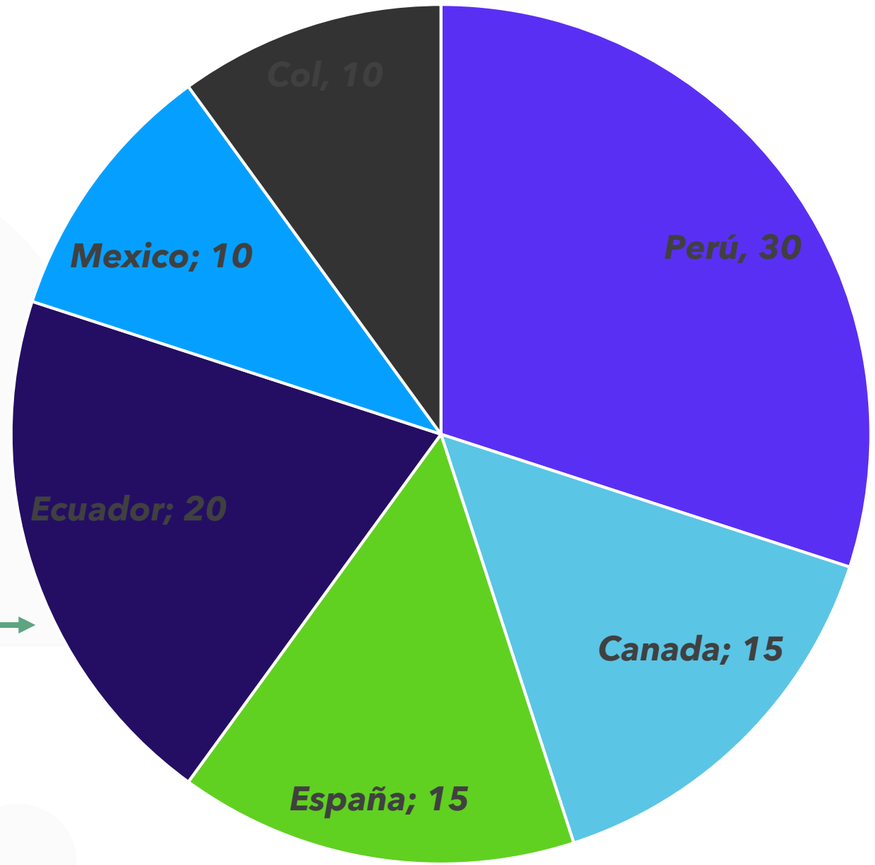
- Un modelo que calcula cualquiera de estas:
  - P(W) or P(w5|w1,w2,w3,w4) se denomina ***modelo de lenguaje***.

Modelos de Lenguaje

Son modelos construidos para predecir la distribución de ***probabilidad*** de la siguiente palabra en una secuencia dadas las palabras anteriores.

Lima es la capital de _____

Contexto

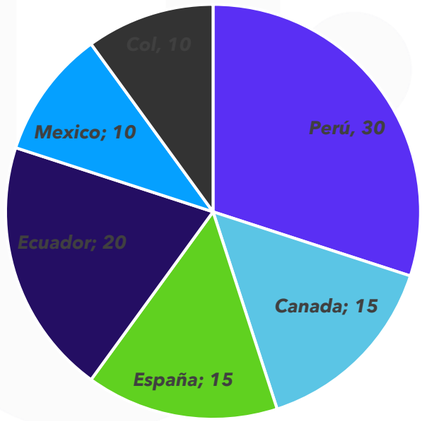


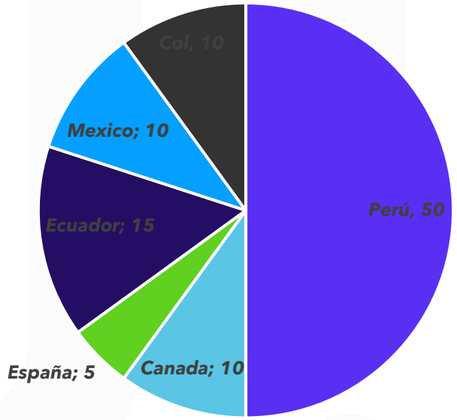
Modelos de Lenguaje (Procesos de Entrenamiento)

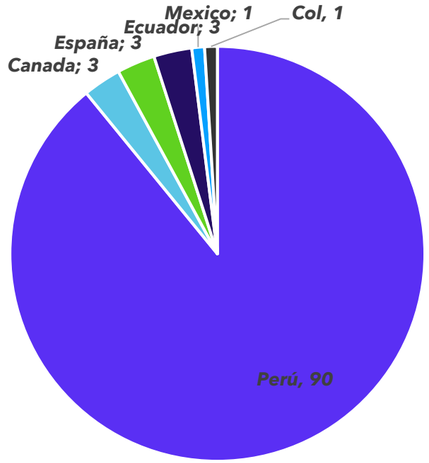
Aprender a crea la ruleta de manera efectiva: ***grandes corpus documentales***

Lima es la ***capital*** de _____

Incremento en el tamaño del conjunto de datos de entrenamiento







Modelos de Lenguaje (Predicción)

Aplico de forma repetitiva el proceso de predecir la palabra siguiente_:

Lima es la ***capital*** de _____

Lima es la ***capital*** de Perú _____

Lima es la ***capital*** de Perú. _____

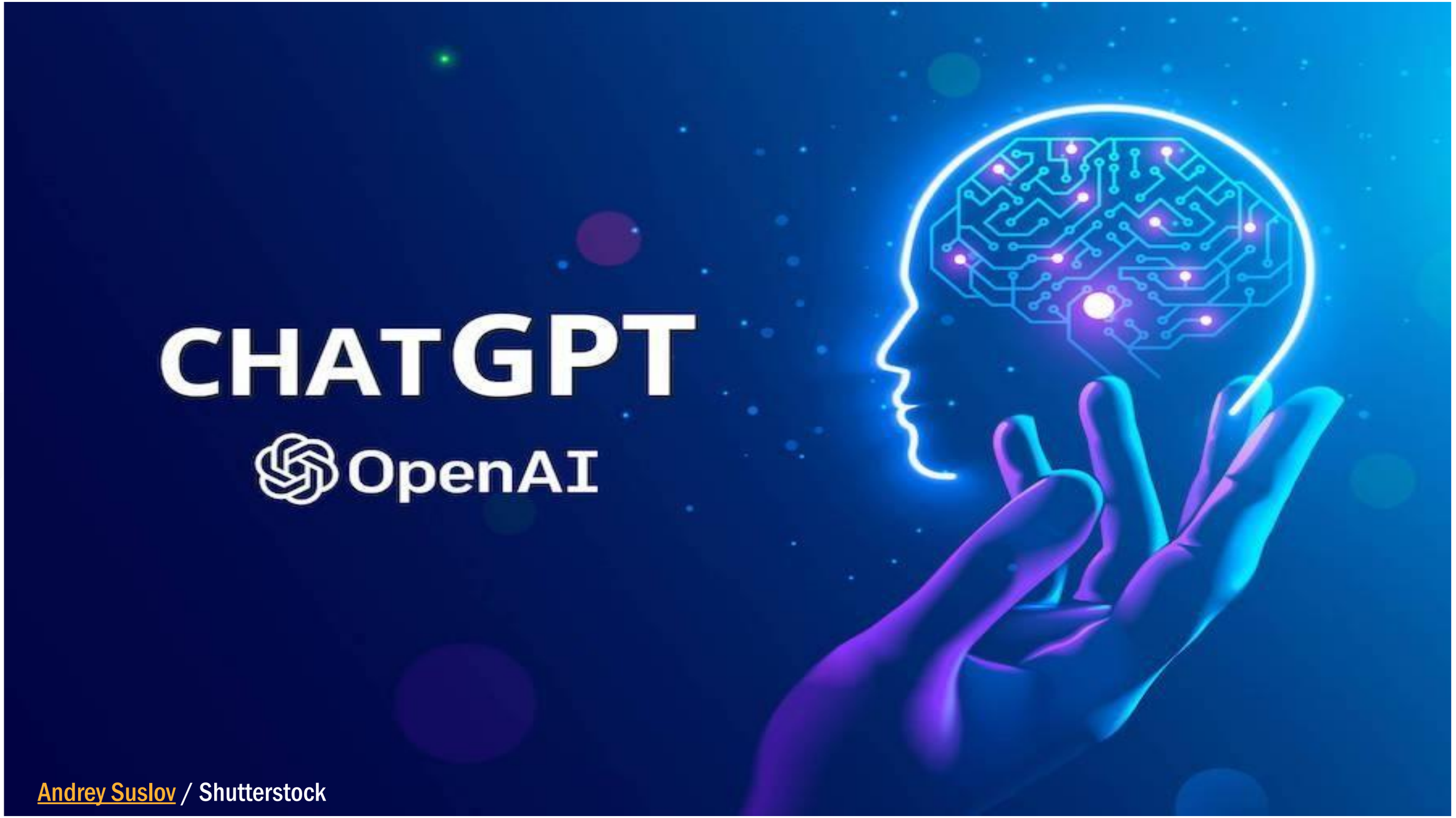
Lima es la ***capital*** de Perú. Se _____

Lima es la ***capital*** de Perú. Se encuentra _____

Lima es la ***capital*** de Perú. Se encuentra situada _____

Lima es la ciudad capital de la República del Perú. Se encuentra situada en la costa central del país, a orillas del océano Pacífico,…

Alucinaciones son palabras o frases generadas por el modelo que carecen de veracidad, sin sentido o son gramaticalmente incorrectas.



Andrey Suslov / Shutterstock

Como calcular P(W)

- Como calcular esta probabilidad conjunta:
  - P(Mugre, que, no, mata, engorda)

- Apliquemos la regla de la cadena de probabilidad:

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

$$P(A,B) = P(A|B)P(B)$$

Recordatorio de la regla de la cadena

- Con más variables
  - P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)

- Formula general:

- $P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

$$P(w_1, w_2 \ldots w_n) = \prod_i P(w_i|w_1, w_2 \ldots w_{i-1})$$

Recordatorio de la regla de la cadena

$$P(w_1, w_2 \dots w_n) = \prod_i P(w_i | w_1, w_2 \dots w_{i-1})$$

P(mugre, que, no, mata, engorda) = P(mugre) × P(que|mugre) × P(no|mugre que) × P(mata|mugre que no) × P(engorda| mugre que no mata)

Como estimar estas probabilidades

- ¿Podemos solo contar y dividir?

$$P(transparent|its\ water\ is\ so) = \frac{Count(its\ water\ is\ so\ transparent)}{Count(its\ water\ is\ so)}$$

- No! Muchas posibles sentencias.
- Nunca vamos a tener suficiente datos para estimarlas.

Suposición de Markov

- Wikipedia: “En teoría de probabilidad y estadística, la propiedad de Markov se refiere a la propiedad de ciertos procesos estocásticos por la cual "carecen de memoria", lo que significa que la distribución de probabilidad del valor futuro de una variable aleatoria depende únicamente de su valor presente, siendo independiente de la historia de dicha variable.”

Suposición de Markov

- Simplificación
  - $P(engorda|\ mugre\ que\ no\ mata) \approx P(engorda|mata)$

- O tal vez:
  - $P(engorda|\ mugre\ que\ no\ mata) \approx P(engorda|no\ mata)$

Suposición de Markov

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

- En otras palabras, se aproxima cada componente en el producto:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

Caso más simple: Modelo Unigramas

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

- Algunas sentencias generadas automáticamente:

Quinto, un , futuro, se, incorpora, el, dinero, es, importante.

Bigramas

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i | w_{i-1})$$

- Algunas sentencias generadas automáticamente:

  afuera, del, carro, se, acuerdo, de, será, un, recuerdo, en, noviembre

Modelos de N-gramas

- Podemos extendernos a trigramas, 4 –gramas o 5-gramas.
- Sin embargo, esto es un modelo de lenguaje insuficiente.
  - Ya que el lenguaje tiene dependencias de largo plazo.
- La mayoría de los correctores ortográficos y modelos de reconocimiento de voz operaban con modelos de 5-gramas.

Dependencias de corto y largo plazo

Al que madruga Dios le _____.
Bogotá es la capital de _____.

Yo crecí en Brasil, pero ahora vivo en Bogotá.
Gracias a eso hablo fluidamente _____ y
_____.

Dependencias a corto y largo plazo.

Estimando las probabilidades de N-gramas

Estimar las probabilidades de bigramas

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

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Notación

MLE: maximum likelihood estimation

Un ejemplo

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$
$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

MLE: maximum likelihood estimation

MLE: Estimación por máxima verosimilitud

- MLE nos permite:
  - Estimar algún parámetro de un modelo M de un conjunto de entrenamiento T
  - Maximiza la probabilidad del conjunto de entrenamiento T dado el modelo M.
- Supongamos que la palabra “amor" aparece 400 veces en un corpus de un millón de palabras.
- ¿Cuál es la probabilidad de que una palabra aleatoria de algún otro texto sea “amor"?
- La estimación de MLE es 400/1,000,000 = .0004

MLE: Estimación por máxima verosimilitud

- Esta puede ser una mala estimación para algún otro corpus.

- Pero es la estimación la que hace más probable que “amor” aparezca 400 veces en un corpus de un millón de palabras.

Conteo de Bigramas

- Consideremos las oraciones del Proyecto de restaurante de Berkeley.
- De 9222 frases

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Conteo de Bigramas

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- Normaliza por unigramas:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Resultado

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Estimación de la probabilidad de una sentencia basados en bigramas

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i | w_{i-1})$$

P(<s> I want english food </s>) = P(I|<s>) × P(want|I) × P(english|want) × P(food|english) × P(</s>|food)

= .000031

Para que nos sirve: Conocimiento del lenguaje

P(english|want) = .0011
P(chinese|want) = .0065
P(to|want) = .66
P(eat | to) = .28
P(food | to) = 0
P(want | spend) = 0
P (i | <s>) = .25

Problemas prácticos

- Hacemos todo en el espacio log
  - Evitamos el desbordamiento
  - Sumar es más rápido que multiplicar

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Toolkit para construir modelos de lenguaje

SRILM

http://www.speech.sri.com/projects/srilm/

KenLM

https://kheafield.com/code/kenlm/

Google N-gram

AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-gram

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
```

Google N-gram: Versión mas reciente 2020

https://storage.googleapis.com/books/ngrams/books/datasetsv3.html

https://books.google.com/ngrams

Evaluación: ¿Qué tan bueno es nuestro modelo?

- ¿Prefiere nuestro modelo de lenguaje las oraciones “buenas” a las “malas”?
  - Asignar mayor probabilidad a oraciones "reales" o "observadas con frecuencia“
- Entrenamos los parámetros de nuestro modelo en un conjunto de entrenamiento.
- Probamos el rendimiento del modelo con datos que no hemos visto.
  - Un ***conjunto de prueba*** es un conjunto de datos invisible que es diferente de nuestro conjunto de entrenamiento, totalmente sin usar.
  - Una ***métrica de evaluación*** nos dice qué tan bien funciona nuestro modelo en el conjunto de prueba.

Evaluación extrínseca de modelos N-grama

- Mejor evaluación para comparar los modelos A y B
  - Pon cada modelo en una tarea
    - corrector ortográfico, reconocedor de voz

- Ejecute la tarea, obtenga una precisión para A y para B
  - Cuántas palabras mal escritas corregidas correctamente
  - Cuantas palabras traducidas correctamente

- Comparar la precisión de A y B

Evaluación extrínseca de modelos N-grama

- Evaluación extrínseca
  - Consume mucho tiempo.

- Se prefiere una evaluación intrínseca: perplejidad
  - Mala aproximación a menos que los datos de prueba se parezcan a los datos de entrenamiento.

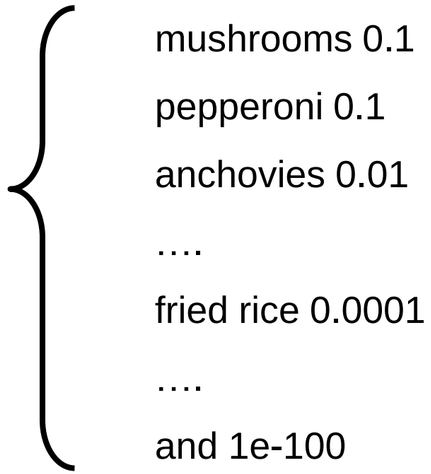
Perplejidad

- Que tan bien puedo predecir la siguiente palabra:

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____



mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

- Los unigramas son terribles en este juego. ¿Por qué?
- Un buen modelo
  - Es aquel que asigna una alta probabilidad a la palabra correcta.

Perplejidad

Wikipedia: In information theory, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. It may be used to compare probability models.... ***Perplexity is sometimes used as a measure of how hard a prediction problem is.***

Perplejidad

- El mejor modelo de lenguaje es aquel que mejor predice una sentencia no vista en el conjunto de prueba:
  - Nos devuelve la más alta P(sentencia)

- Perplejidad es el inverso de la probabilidad del conjunto de prueba normalizado por el número de palabras:

$$PP(S) = P(s_1 s_2 \ldots s_m)^{-\frac{1}{m}} = \sqrt[m]{\frac{1}{P(s_1 s_2 \ldots s_m)}}$$

Minimizar la perplejidad es lo mismo que maximizer la probabilidad.

BIGRAMAS

$$PP(W) = \sqrt[m]{\prod_{i=1}^{m} \prod_{j=1}^{|s_i|} \frac{1}{P(w_j^{(i)} | w_{j-1}^{(i)})}}$$

Aproximación de Markov

Perplejidad mini ejemplo

- Suponga una sentencia compuesta de dígitos aleatorios (1 2 3 9). (palabra=digito para este ejemplo).

- Supongan un modelo de lenguaje que asigna una probabilidad de P=1/10 a cada palabra/dígito. ¿Cuál es la probabilidad de un test set de N sentencias compuestas por una sola palabra?.

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

Generalización y Zeros

El método de visualización de Shannon

- Elige un bigrama al azar (<s>, w) según su probabilidad
- Ahora elige un bigrama aleatorio (w, x) de acuerdo a su probabilidad
- Y así sucesivamente hasta que elijamos </s>
- Luego une las palabras

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Aproximación a las obras de Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Corpus Shakespeare

- N=884,647 tokens, V=29,066.

- Shakespeare produjo 300.000 tipos de bigramas de 844 millones de bigramas posibles.
  - Cerca del 99,96% del conjunto posible bigramas nunca son vistos.

- Empeora con los cuatrigramas.

- Set de entrenamiento:
  - “Hola mundo”, “Hola <NAME>”, “Hola amigo”, “Hola compañero”
- Set de prueba:
  - “Hola parcero”
- ***P(“parcero”|”hola”) = 0 PROBLEMA DE GENERALIZACIÓN (DIVISIÓN POR CERO)***

Laplace Smoothing (Suma 1)

- Imagina que vimos cada palabra una vez más de lo que lo hicimos.
- ¡Solo agregue uno a todos los conteos!
- Estimación de MLE:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Laplace:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Laplace Smoothing (Suma 1)

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace Smoothing (Suma 1)

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

P*(want|i)=827+1/2533+1446=0.21

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

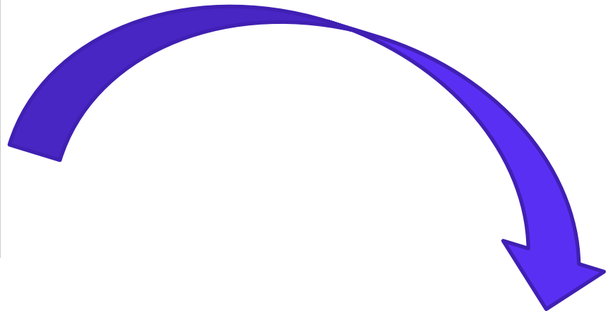
Observe la suma 1 en la tabla

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Suponga un Vocabulario de 1446 Term.

Laplace Smoothing (Suma 1)

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0



	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Referencias

- Jurafsky D. and Martin J. (2021) Speech and Language Processing (3rd ed. draft). Online: https://web.stanford.edu/~jurafsky/slp3/
- (Chapter 2: Speech and Language Processing)

- Yoav Goldberg (2017). Neural Network Methods in Natural Language Processing.

- In Deng, L., & In Liu, Y. (2018). Deep learning in natural language processing.