Evaluación de modelos de lenguaje

Modelos de generación de texto

- Entrada: Texto
- Salida: Palabra (siguiente)
- Model: aprende P(Palabra | Texto)
- Ejemplo
  - Vocabulario = {Yo, al, perro, gato, paseo}
  - P(Palabra | “Yo paseo al”) =
    - P(Yo | “Yo paseo al”) = 0.1
    - P(Al | “Yo paseo al”) = 0.0
    - P(Perro | “Yo paseo al”) = 0.4
    - P(Gato | “Yo paseo al”) = 0.4
    - P(Paseo | “Yo paseo al”) = 0.1

Ejemplo de generación determinística

- Entrada inicial: “”
  a. Buscar palabra que maximice P(palabra | “”)
     - “Hola”
  b. Buscar palabra que maximice P(palabra | “Hola”)
     - “cómo”
  c. Buscar palabra que maximice P(palabra | “Hola cómo”)
     - “estás”
  d. Buscar palabra que maximice P(palabra | “Hola cómo estás”)
     - FIN_DE_ORACIÓN
- Salida: “Hola cómo estás”

Ejemplo de generación no determinística

- Entrada inicial: “”
  a. Elegir palabra según distribución P(palabra | “”)
     - P(“Hola” | “”) = 0.2
     - P (“cómo” | “”) =0.1
     - ..
       - “Hola”
  b. Elegir palabra según distribución P(palabra | “Hola”)
     - (idem…) → “cómo”
  c. Elegir palabra según distribución P(palabra | “Hola cómo”)
     - P(“andás” | “”) = 0.3
     - P (“estás” | “”) =0.2
       - “estás”

Ejemplo de generación determinística con texto inicial

- (Mismo modelo que antes)
- Entrada inicial: “Chau”
  a. Buscar palabra que maximice P(palabra | “Chau”)
     - “hasta”
  b. Buscar palabra que maximice P(palabra | “Chau hasta”)
     - “mañana”
  c. Buscar palabra que maximice P(palabra | “Chau hasta mañana”)
     - FIN_DE_ORACIÓN
- Salida: “Chau hasta mañana”

Modelos de n-gramas

- ¿Qué longitud puede tener el texto de entrada?
  - ¿Cuánto mira el modelo hacia el pasado?
- n-grama: secuencia de n palabras en orden
  - hola: 1 grama
  - hola-como: 2 grama
  - hola-como-estas: 3 grama
- Modelos *clásicos*
  - Basados en “contar” las apariciones de las gramas

Generación determinística - modelo de 2 gramas

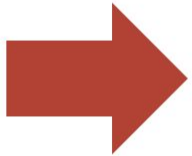
- Entrada inicial: “NADA NADA”
  a. Buscar palabra que maximice P(palabra | “NADA NADA”)
     - “Hola”
  b. Buscar palabra que maximice P(palabra | “**NADA** Hola”)
     - “cómo”
  c. Buscar palabra que maximice P(palabra | “**Hola** cómo”)
     - “estás”
  d. Buscar palabra que maximice P(palabra | “**cómo** estás”)
     - FIN_DE_ORACIÓN
- Salida: “Hola cómo estás”
- Solo ve 2 palabras en el pasado
- Mayor n → más cómputo

Modelos de Redes Neuronales

- Estado = Vector n dimensional de estado previo
- Inicializar $ESTADO_0$
  a. Buscar palabra que maximice P(palabra | $ESTADO_0$)
     - “Hola”
  b. Buscar palabra que maximice P(palabra | $ESTADO_1$)
     - “cómo”
  c. Buscar palabra que maximice P(palabra | $ESTADO_2$)
     - “estás”
  d. Buscar palabra que maximice P(palabra | $ESTADO_3$)
     - FIN_DE_ORACIÓN
- Salida: “Hola cómo estás”
- $ESTADO_i$: vector D dimensional que codifica estado previo

Codificación de palabras *One-hot*

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

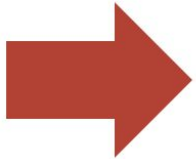


	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

- Problemas
  a. Palabras parecidas no tienen representación similar
  b. Difícil escalar a vocabularios grandes

Codificación de palabras mediante *embeddings*

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

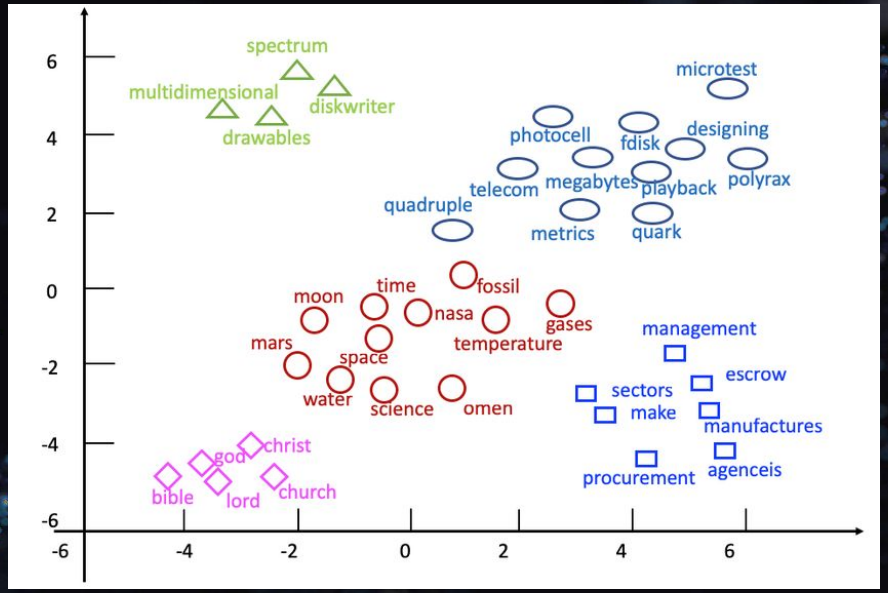


	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

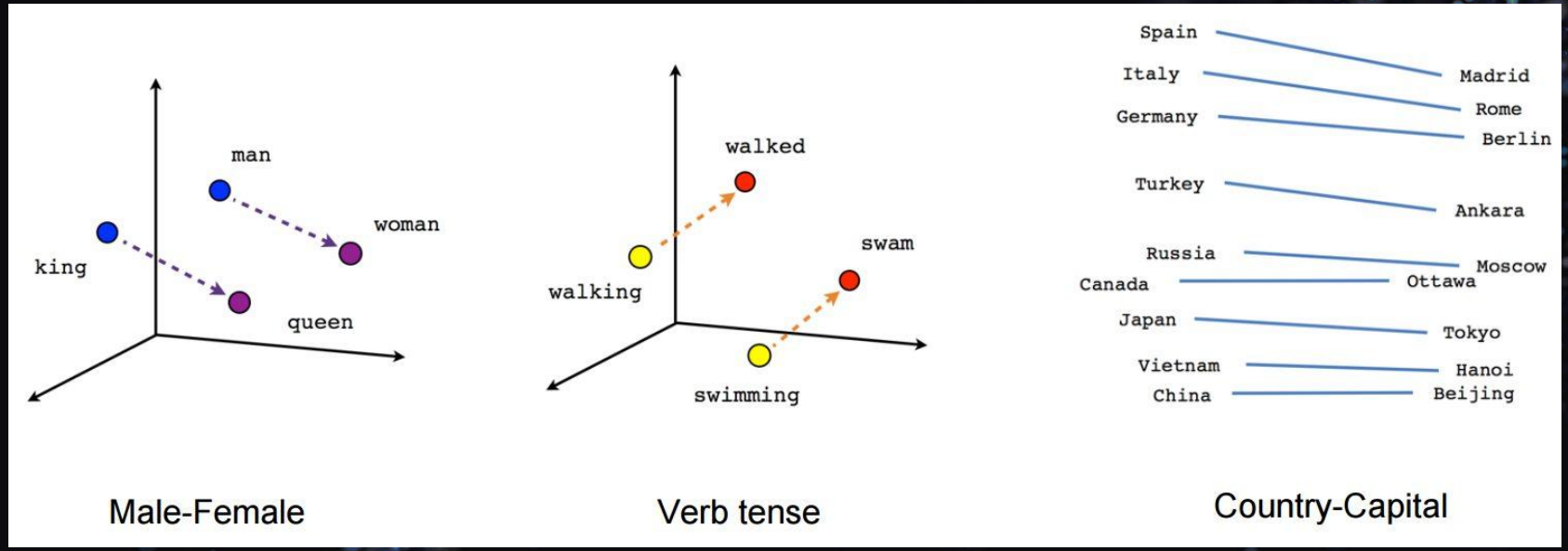
- (Embedding diseñado a mano)
- Ventaja: solos 3 features, palabras similares → vec similar

Codificación de palabras mediante *embeddings*

- Ejemplo real, codificación reducida a 2 dimensiones
- Embedding aprendido por una red neuronal entrenada de forma no supervisada



Codificación de palabras mediante *embeddings*



- Embedding aprende relaciones desde los datos

Modelos de lenguaje

- A nivel de palabras
  - P(palabra | “Hola”)
  - Ventaja: Mejor desempeño/menor coste comp.
  - Desventaja: Vocabularios muy grandes
- A nivel de caracteres
  - P(carácter | “Hol”)
  - Ventaja: Vocabulario pequeño
  - Desventaja: Puede generar palabras inexistentes

Tareas con modelos de lenguajes

- Modelos de lenguaje
- Se utilizan o son parte de:
  - Generadores de texto
  - Clasificadores de texto
  - Resúmenes automáticos
  - Chatbots
  - Sistemas de pregunta y respuesta
  - Traducción
  - Extracción de palabras clave

Evaluación de modelos de Lenguaje

Diversas métricas según tarea

- Generación de texto
  - Perplejidad
- Traducción
  - Word Error Rate (WER)
  - BiLingual Evaluation Understudy (BLEU)

Perplejidad (PP o Perplexity)

- Calcula probabilidades del modelo
  - Sobre conjunto de prueba
- Ejemplo
  - Texto = “Hola como estás”
  - Calcular P(“$w_1 w_2 w_3 \ldots w_n$ ”)
  - P(“Hola cómo estás”) = P(“Hola” | “”) * P(“cómo” | “Hola”) * P(“estás” | “Hola cómo”)
    - Modelos distintos asignan distinta P
    - Mayor probabilidad → mejor modelo

Perplejidad (PP o Perplexity)

- Perplejidad = $1/P(\text{“}w_1 w_2 w_3 \ldots w_n\text{”})$
  - Valores: 1 a infinito
  - Si $P \rightarrow 0$, $PP \rightarrow$ infinito
  - Si $P \rightarrow 1$, $PP \rightarrow 1$
- Problema: PP depende de n
  - n grande $\rightarrow$ P chica $\rightarrow$ PP grande
- Perplejidad = $(1/P(\text{“}w_1 w_2 w_3 \ldots w_n\text{”}))^{1/n}$
  - Raiz enésima
  - Normaliza por longitud del texto

Perplejidad (PP o Perplexity)

- Ejemplos
  - Modelo de 2-grama
  - Vocabulario = {A, B, C}
- PP(B A) = $[\ 1/\ (\ P(B) * P(A|B))\ ]^{½}$
  $= [1/\ (\ 0.25 * 0.25)\ ]^{½}$
  $=[\ 8\ ]^{½} = 2.82$
- PP(A C B A) = $[\ 1/\ (\ P(A) * P(C|A) * P$
  $P(A|B))\ ]^{¼}$
  $= [1/\ (0.25*0.3*0.25*0.25)\ ]^{¼}$
  $= (213.3)^{¼} = 3.82$

w_1	$P(w_1)$
A	0.25
B	0.25
C	0.5

w_1	w_2	$P(w_1\|w_2)$
A	A	0.3
B	A	0.3
C	A	0.3
A	B	0.25
B	B	0.5
C	B	0.25
A	C	0.6
B	C	0.3
C	C	0.1

Traducción de texto

- Original: “I have been blessed”
  - Referencia: “He sido bendecido”
  - Predicción: “He recibido una bendición”
- Comparar referencia contra predicción
  - Similar en contenido
  - Similar el longitud

Word Error Rate (WER)

- Basada en distancia Levenshtein o Edit
- Distancia entre strings:
  - # de cambios **mínima** para llegar de uno al otro
- Tipos de **cambios**
  - Inserciones
  - Borrados
  - Sustituciones

Word Error Rate (WER)

- Ejemplo
  - Referencia: “He sido bendecido”
  - Predicción: “He recibido una bendición”
- 3 cambios
  - He **recibido** una bendición
  - He sido **una** bendición
  - He sido **bendición**
  - He sido bendecido
- WER = 3

Word Error **Rate** (WE**R**)

- WE**R** = #Cambios
  - Depende de la longitud de las oraciones
  - Normalizar
- WER = #Cambios / #Palabras
  - #Palabras de la **referencia** (¿por qué?)
- Ejemplo
  - Referencia: “He sido bendecido”
  - Predicción: “He recibido una bendición”
  - WER = 3/3 = 1

Word Error Rate (WER): Ejemplos

- Ejemplo #2
  - Referencia: “He sido bendecido”
  - Predicción: “Me han bendecido”
  - 2 sustituciones
  - WER = 2/3
- Ejemplo #3
  - Referencia: “He sido bendecido”
  - Predicción: “Este modelo no debe ser muy bueno”
  - 3 sustituciones, 4 borrados
  - WER = 7/3 = 2.3

BLEU

- Compara n-gramas entre
  - referencias: traducciones hechas por humanos
  - predicción: salida del modelo
- Basado en *Precision*
  - Importa que las palabras predichas estén en la referencia
  -

BLEU básico

- Referencia = $r_1 r_2 r_3 \ldots r_N$
- Predicción = $p_1 p_2 p_3 \ldots p_M$
- $BLEU_1 = A/M$
  - A = # aciertos de la predicción
- Ejemplo:
  - Referencia: **El gato está afuera**
  - Predicción: **El gato fue al exterior**
    - A = 2
    - M = 5
    - BLEU = 2/5

BLEU con repetidos

- $BLEU_1$ = A/M
  - A = #aciertos de la predicción
- Referencia: **el gato está en el patio**
- Predicción: **el el el el**
  - A = 2 (# de veces que aparece **el** en la referencia)
  - M = 4
  - BLEU = 2/4 = 0.5

BLEU con varias referencias

- Traducciones varían
- Evaluar con más de una referencia
- $BLEU_1 = A/M$
  - A = # aciertos de la predicción en **cualquier** referencia
- Ejemplo:
  - Referencia 1: **El gato está afuera**
  - Referencia 2: **El gato está en el exterior**
  - Predicción: **El gato fue al exterior**
    - A = **3**
    - M = 5
    - BLEU = 3/5

BLEU con n gramas

- El orden importa
- $BLEU_1$ no lo considera
- Ejemplo:
  - Referencia 1: **El gato está afuera**
  - Predicción: **afuera gato esta el**
    - A = **4**
    - M = 4
    - BLEU = 4/4
- Utilizar coincidencia de n-gramas

$BLEU_k$ con k gramas

- $BLEU_k = A / M$
  - A = #k-gramas que acierta la predicción
  - M = #k-gramas posibles de la predicción
- Ejemplo con $BLEU_2$:
  - Referencia 1: **El gato está afuera**
  - Predicción: **afuera gato está el**
    - A = **1 (gato está)**
    - M = 3 (afuera-gato, gato-esta, esta-el
    - $BLEU_2 = 1/3$

Problema: BLEU con M chico

- $BLEU_1 = A/M$
  - A = #aciertos de la predicción
- Referencia: el gato está en el patio
- Predicción: “”
  - A = 0
  - M = 0
  - BLEU = 0/0 = ¿?
- Predicción: “el”
  - A = 1, M = 1
  - BLEU = 1/1 = 1

BLEU con penalización por brevedad

- $BLEU_1 = (A/M) * BP$
  - A = #aciertos de la predicción
  - BP = Brevity Penalty (Penalización por Brevedad)
  - BP = complicado
    - depende del conjunto de texto a predecir
    - Cuanto más corta la oración (relativa al resto)
      - Mayor la penalización