Modelos de Lenguaje

- N-grams
- Neurales

Modelos de Lenguaje

¿Cuál es la probabilidad de "gato" en:

Cuando llegue a casa me tengo que acordar de darle de comer al _____

Modelos de Lenguaje

¿Cuál es la probabilidad de "gato" en:

Cuando llegue a casa me tengo que acordar de darle de comer al _____

¿Es más probable "gato" o "perro"?

Modelos de Lenguaje

¿Cuál es la probabilidad de "gato" en:

Cuando llegue a casa me tengo que acordar de darle de comer al _____

¿Es más probable "gato" o "perro"?

Aplicaciones:

- *Traducción*
- *Reconocimiento de habla*
- *Corrección en la escritura*
- *Sistema de ayuda*

Modelos de Lenguaje

¿Podemos estimar la probabilidad a posteriori de una palabra dada su historia?

p(w | h) = probabilidad de palabra *w* dada su historia *h*

N-grams

Podemos calcularlo como:

$$p(w \mid h) = \#(h , w) / \#(h)$$

$$p(\text{"gato"} \mid \text{"Cuando llegue a casa me tengo que acordar de darle de comer al"}) = \frac{\#(\text{"Cuando llegue a casa me tengo que acordar de darle de comer al gato"})}{\#(\text{"Cuando llegue a casa me tengo que acordar de darle de comer al"})}$$

N-grams

Podemos calcularlo como:

$$p(w \mid h) = \#(h , w) / \#(h)$$

$$p(\text{"gato"} \mid \text{"Cuando llegue a casa me tengo que acordar de darle de comer al"}) = \frac{\#(\text{"Cuando llegue a casa me tengo que acordar de darle de comer al gato"})}{\#(\text{"Cuando llegue a casa me tengo que acordar de darle de comer al"})}$$

Pero cómo cuento cuántas veces aparece una frase en **el lenguaje**.

Además, el lenguaje es creativo, es probable que (***h,w***) *no exista*

N-grams

Si lo calculamos con regla de la cadena:

$$p(w_1 w_2 w_3 ... w_n) = p(w_1) * p(w_2|w_1) * p(w_3|w_1 w_2) * p(w_4|w_1 w_2 w_3) * \dots * p(w_n|w_1 w_2 w_3 \dots w_{n-1})$$

p("Cuando llegue a casa me tengo que acordar de darle de comer al gato") =
p(cuando) * p(llegue | Cuando) * p(a | Cuando llegue) * … * p(gatogato | Cuando llegue ... a ...)

$$p(w_1^n) = \prod_{k=1}^{N} p(w_k \mid w_1^{k-1})$$

N-grams

Si lo calculamos con regla de la cadena:

$$p(w_1 w_2 w_3 ... w_n) = p(w_1) * p(w_2|w_1) * p(w_3|w_1 w_2) * p(w_4|w_1 w_2 w_3) * \dots * p(w_n|w_1 w_2 w_3 \dots w_{n-1})$$

p("Cuando llegue a casa me tengo que acordar de darle de comer al gato") =
p(cuando) * p(llegue | Cuando) * p(a | Cuando llegue) * … * p(gatogato | Cuando llegue ... a ...)

$$p(w_1^n) = \prod_{k=1}^{N} p(w_k \mid w_1^{k-1})$$

Problema: no soluciona el problema de que no exista una determinada cadena de palabras

N-grams

N-grams

Si lo calculamos con regla de la cadena:

$$p(w_1 w_2 w_3 ... w_n) = p(w_1) * p(w_2|w_1) * p(w_3|w_1 w_2) * p(w_4|w_1 w_2 w_3) * \dots * p(w_n|w_1 w_2 w_3 \dots w_{n-1})$$

p("Cuando llegue a casa me tengo que acordar de darle de comer al gato") =
p(cuando) * p(llegue | Cuando) * p(a | Cuando llegue) * … * p(gatogato | Cuando llegue ... a ...)

$$p(w_1^n) = \prod_{k=1}^{N} p(w_k \mid w_1^{k-1})$$

Problema: no soluciona el problema de que no exista una determinada cadena de palabras
Ventaja : podemos buscar **n** más chico que permita calcular las probabilidades y aproximar la probabilidad

p(gato | Cuando llegue a casa me tengo que acordar de darle de comer al) ~ p(gato | darle de comer al)

5-gram

$$p(w_1^n) \sim \prod_{k=1}^{N} p(w_k \mid w_{n-N+1}^{k-1})$$

N-grams

Estimador de Máxima Verosimilitud (MLE):

$$p(w_1^n) \sim \prod_{k=1}^{N} p(w_k \mid w_{n-N+1}^{k-1}) \longrightarrow P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

Evaluación de LM

Evaluación extrínseca: Usar los resultados de nuestro n-gram en alguna aplicación externa

- Entrenar modelos para después testearlos en otra aplicación puede ser costoso

Evaluación intrínseca: Evaluar el modelo en un test set

- Es importante que las oraciones de test no estén en el training set
- Para evaluar usamos ***perplexity***

Perplexity

Inversa de la probabilidad en un set de testeo, normalizada por la cantidad de palabras

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

$$\begin{aligned}\mathrm{Perplexity(C)} &= \sqrt[N]{\frac{1}{\prod_{i=1}^{m} p(s_i)}} \\ &= 2^{\log_2 [\prod_{i=1}^{m} p(s_i)]^{-N}} \\ &= 2^{-\frac{1}{N}\log_2 [\prod_{i=1}^{m} p(s_i)]} \\ &= 2^{-\frac{1}{N}\sum_{i=1}^{m} \log_2 p(s_i)}\end{aligned}$$

La perplexity solo es comparable si dos modelos fueron entrenados con el mismo corpus

- Esto se debe al tratamiento que se le da a los <UNK>
- Si tenemos vocabulario chico y asignamos alta probabilidad a <UNK>, la PP va a ser muy alta

Detalles de implementación

- Usar log probability
- Marcar inicio y final de oración (<s> </s>)
- Cuidar el género de los textos
- Cuidar los dialectos de los textos
- Tratamiento de las palabras no vistas en el train
  - Marcar como <unk> en traning todas las que tienen muy baja frecuencia
  - Tratar a todos los <unk> igual
- Tratamiento de probabilidades = 0 (smoothing)

Smoothing

Training set:

```
Cuando llegue a casa me tengo que acordar de darle de comer al gato
El perro me comió la tarea
Me olvidé de darle de comer al canario
```

Test set:

```
Cuando llegue a casa me tengo que acordar de darle de comer al perro
```

p(perro | darle de comer al) = ?

Smoothing: redistribución de las probabilidades

Smoothing

Laplace smoothing (add-one)

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

Add-K smoothing

$$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1}) + kV}$$

Backoff and Interpolation: usando menos contexto (n mas chico) cuando no existe el n-grama

¿Cómo generalizamos en N-gram?

Entrenamiento: "Cuando llegue a casa me tengo que acordar de darle de comer al gato"

¿Cuál es la probabilidad 5-gram de "perro" dada la siguiente frase?

Testeo: "Me estaba yendo a dormir y justo me acordé de darle la comida al ___"

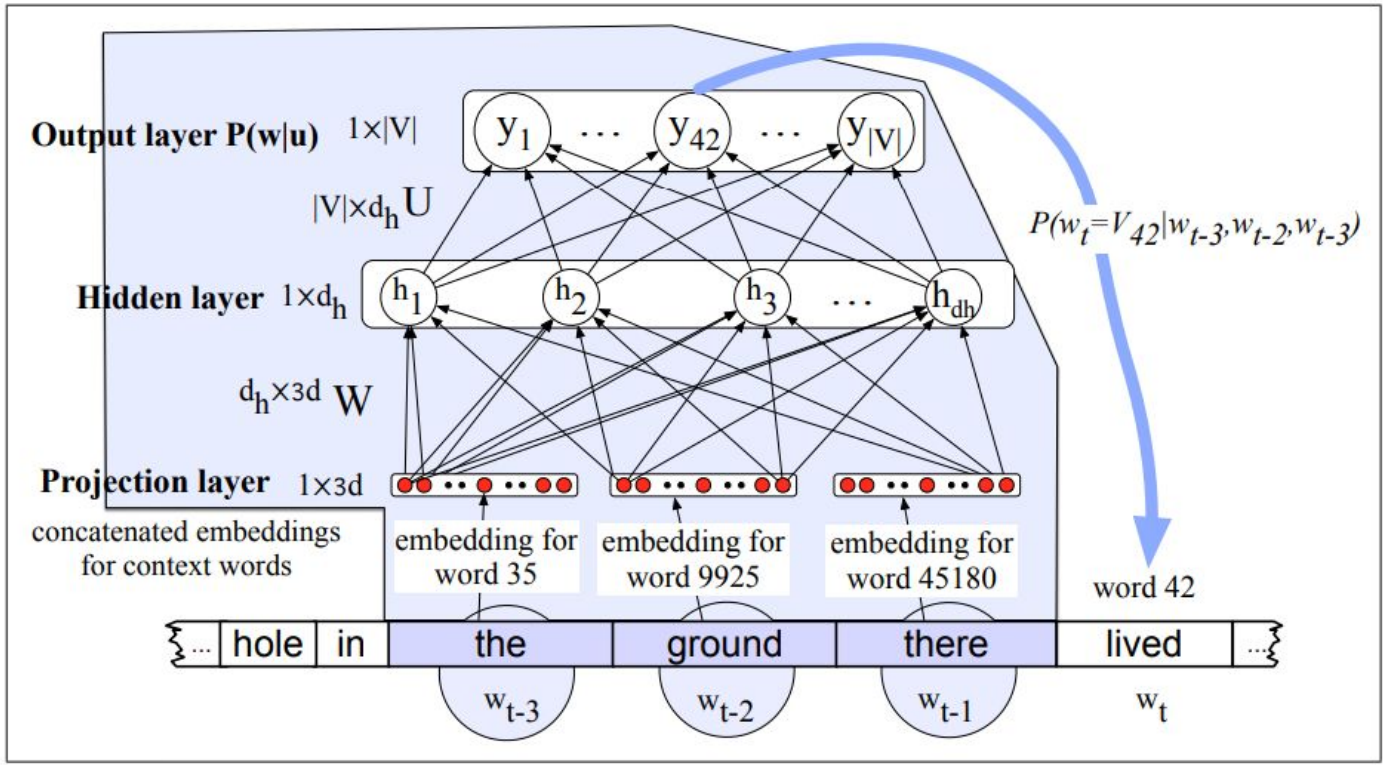
N-gram no puede generalizar a palabras similares.

Los embeddings y las redes neuronales (feed forward) nos pueden ayudar

Modelos de Lenguaje con redes

(Neural Language Models)

Neural LM

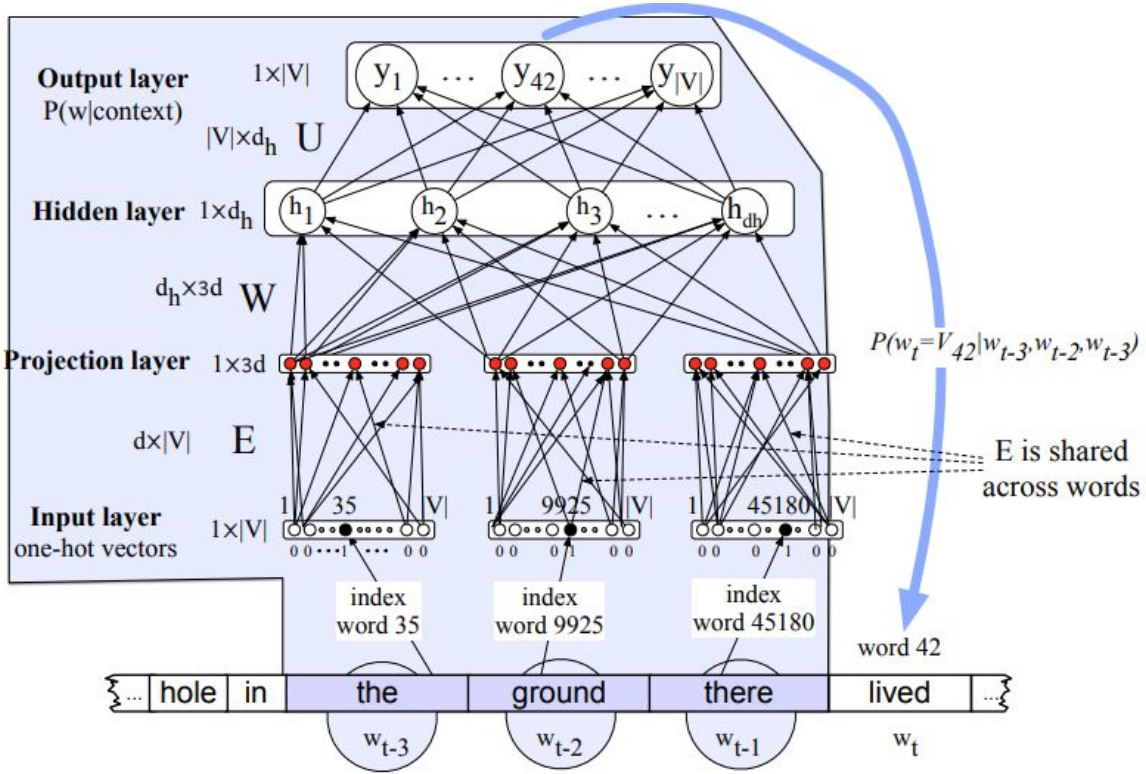


Podemos usar embeddings pre-entrenados

Estos embeddings ya tienen codificadas las relaciones semánticas

Pero no están calculados para optimizar la tarea para la cual creamos la red (ej. Predicción)

Neural LM



Para mejorar nuestra red podemos agregarle una capa donde se calcule el embedding.

La primera capa va a tomar los one-hot encodings de las palabras, pasa por la capa de embedding y luego a la capa oculta.

El backpropagation incluye a los embeddings, por lo que se calcular teniendo en cuenta la tarea