Big Data y Minería de Datos

Unidad 4. Procesado del Lenguaje Natural

U4.2. Representación del lenguaje

Autores: José Antonio Ruipérez Valiente (jruiperez@um.es),
Mariano Albaladejo González (mariano.albaladejog@um.es) y
Manuel Jesús Gómez Moratilla (manueljesus.gomezm@um.es)

UNIVERSIDAD DE
MURCIA

Unidad 4. Procesado del Lenguaje Natural
Estructura

UNIVERSIDAD DE MURCIA

- Procesado básico de texto
- Normalización del texto
- Bag-of-words y TF-IDF
- N-gramas

Big Data y Minería de Datos
Contenidos

UNIVERSIDAD DE MURCIA

Introducción a la minería de datos

Técnicas de inteligencia artificial

Reglas
Fundamentos lógicos
Sistemas basados en reglas
Reglas de asociación

Aprendizaje computacional
Fundamentos
Regresión y clasificación
No supervisado

Procesado del lenguaje natural
Fundamentos
Representación y tópicos
Sentimientos y NER

Introducción al big data

UNIVERSIDAD DE MURCIA

Unidad 4. Procesado del Lenguaje Natural

PROCESADO BÁSICO DE TEXTO

- Cualquier tarea de PLN debe hacer las siguientes tareas:
  - Segmentar/tokenizar palabras del texto
  - Normalizar el formato de las palabras
  - Segmentar las frases del texto
- Esto lo haremos sobre un corpus (colección) de texto/audios
- ¿Cuántas palabras hay en "*Vamos a comer, niños.*"?
- El tratar las comas "," y los puntos ".", y otros signos de puntuación, dependerá de la tarea
- Veamos ahora la siguiente respuesta a una pregunta:
  - "*Yo gestiono uhmm, princ- principalmente los servidores*"
  - Hay dos disfluencias, la primera "*uhmm*" es un relleno (*filler*) y la segunda "*princ-*" es un fragmento
  - ¿Son palabras? Dependerá de la aplicación
- ¿Qué haremos con las mayúsculas? Misma situación

- ¿Que vemos entre “gato”, “gata” y “gatitos”?
  - Una misma raíz (gat) con distintos morfemas y las tres palabras son parte de las formas que puede adquirir
  - Los lemas mantendrán la misma raíz y sentido
  - La forma de las palabras que se derivan de su raíz
- ¿Cuántas palabras hay en el castellano? Distinguir entre:
  - Los tipos distintos de palabras en el vocabulario V, extraído de un cuerpo, con cantidad $|V|$ de tipos de palabras
  - El número de palabras (tokens) con valor N
- Analiza estos valores en la frase “*El gatito de María estaba jugando con el ovillo de lana el día de ayer*”

	Tokens = N	Tipos = |V|
Switchboard phone conversations	2.4 millones	20 mil
Shakespeare	884 mil	31 mil
Google N-grams	1 trillones	13 millones

El corpus

- Texto aparece con uno o más escritores con un lenguaje y propósito específico
- La dimensión más importante es el lenguaje (hay más de 7000)
- Es importante probar algoritmos/métodos en varios idiomas
  - Principalmente se trabaja en inglés (también en español)
- Code switching (o cambio de código) – Múltiples idiomas
- Generación de hoja de datos del corpus:
  - Motivación: ¿Quién y por qué se recolectó?
  - Situación: En qué contexto (tarea, conversación, etc)
  - Variedad del lenguaje: ¿Qué lenguaje/dialecto aparece?
  - Demografía del hablante: ¿Cuál es la procedencia, sexo y origen?
  - Proceso de recolección: Instrumentos, procesado, metadatos…
  - Proceso de anotación: Anotaciones y de qué tipo
  - Distribución: Copyright o propiedad intelectual

UNIVERSIDAD DE MURCIA

Unidad 4. Procesado del Lenguaje Natural

NORMALIZACIÓN DEL TEXTO

Tokenización de palabras

- Proceso de segmentar el texto en palabras
- Normalmente querremos mantener puntuación/expresiones
  - Puntuación (.,), caracteres (%$), fechas (01-02-22), URL, acrónimos (PhD), hashtag…
- También podría unificar palabras en múltiples expresiones
- Estándar Peen Treebank para tokenizar es ampliamente usado

Input: "The San Francisco-based restaurant," they said, "doesn't charge $10".

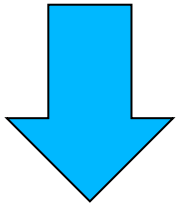
Output: "␣The␣San␣Francisco-based␣restaurant␣,␣"␣they␣said␣,␣ "␣does␣n't␣charge␣$␣10␣"␣.

- Debe ser muy rápido, se usan algoritmos deterministas basados en ER con autómatas finitos
- Proceso más complicado en otras lenguas como el Chino y se suelen usar caracteres en vez de palabras

- Poner palabras/tokens en un formato estándar, aunque se pierde algo de información (guay, guai, guachi)
- Unificar capitalización es otro ejemplo de normalización
  - En algunas aplicaciones es útil, por ej. sentimientos
- Lematización para determinar la raíz de las palabras (soy, eres, somos) a el mismo lema "es"
  - La frase "*Él estaba leyendo historias de detectives*" sería lematizada como "*El estar leer historia detective*"
  - Se hace mediante parseo morfológico de la palabra
- La morfología estudia la forma en la que las palabras son construidas mediante morfemas:
  - Estema: Morfema raíz principal de la palabra
  - Afijos: Significados adicionales de varios tipos
  - "Gatitos" a "gat" y "itos"

- Los algoritmos de lematización son complejos
- Se usa un método más sencillo que es cortar los afijos finales
- Esta forma ingenua morfológica se llama estemización
- El algoritmo es un ampliamente usado

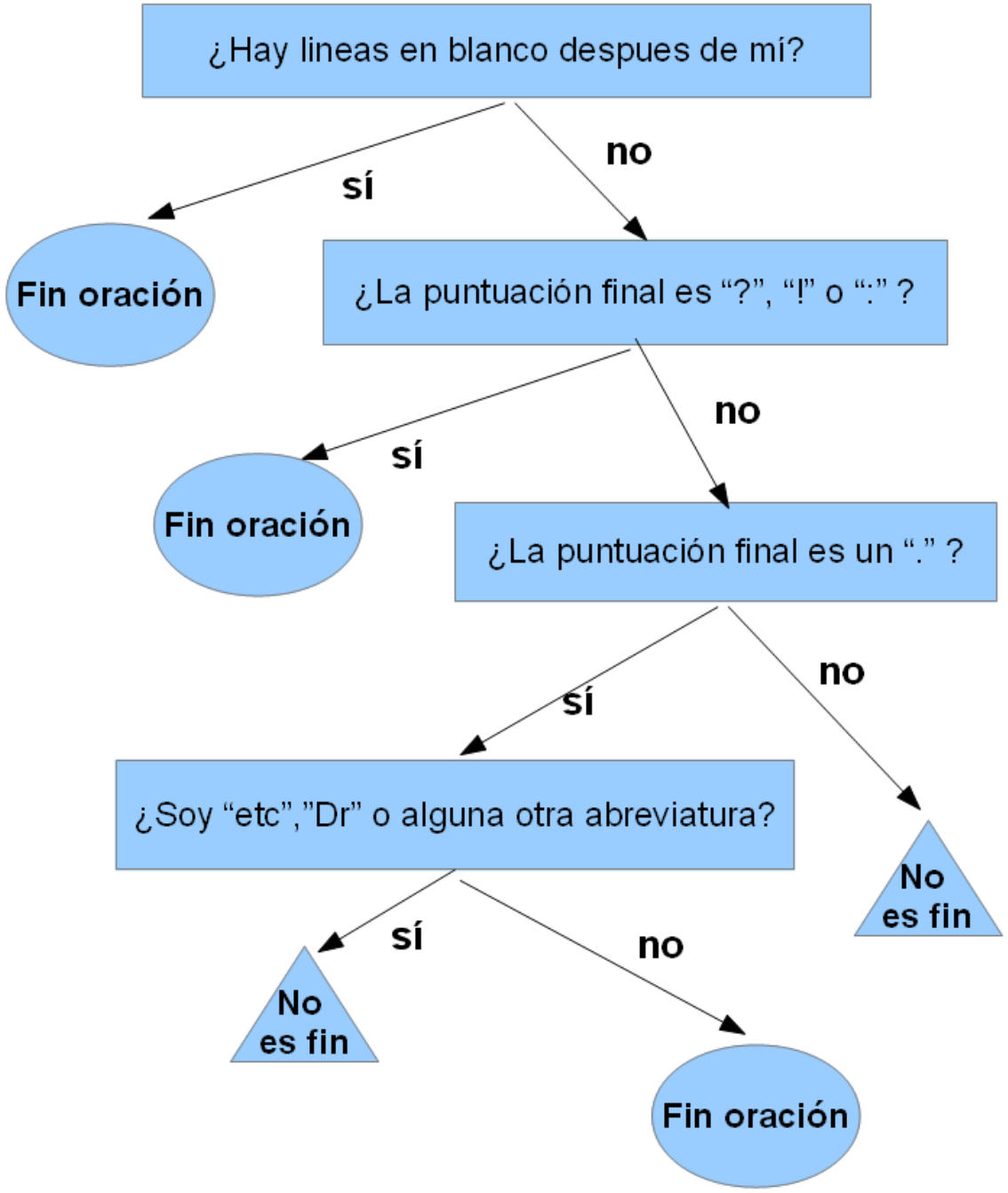
This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.



Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note

- Uso de la puntuación para este proceso
  - Las marcas de pregunta y exclamación no son ambiguas
  - Los puntos pueden serlo más, por ejemplo Sr. o Inc.
  - Inc. Podría incluso ser final de frase y de abreviación
  - Por ello, a veces se hace conjuntamente con la tokenización
- Objetivo: Decidir si un punto es final de frase o abreviación
  - Uso de reglas, aprendizaje computacional, o diccionarios
- Por ejemplo, Stanford CoreNLP, se aplica la regla, después de la tokenización
  - Si la frase termina en ".", "!", o "?" y no ha sido agrupada (por ej, con número o abreviación) entonces termina.

Unidad 4. Procesado del Lenguaje Natural
Ejemplo de árbol de decisión para segmentar frase



¿Cómo de similares son dos cadenas?

- En corrección de palabras. Si el usuario escribió “aselrgia” ¿cuál es la más cercana: alergia, alegría, alergias?
- En biología para el alineamiento dos secuencias de nucleótidos

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

- Alineamiento resultante:

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

- También se usa para la traducción computacional, extracción de información, reconocimiento de habla…
- Transformación de uno a otro con tres operaciones: Inserción, eliminación y sustitución

Unidad 4. Procesado del Lenguaje Natural
Ejemplos

UNIVERSIDAD DE MURCIA

- En corrección de palabras. Si el usuario escribió “aselrgia” ¿cuál es la más cercana: alergia, alegría, alergias?

```
I  N  T  E  *  N  C  I  Ó  N
|  |  |  |  |  |  |  |  |  |
*  E  J  E  C  U  C  I  Ó  N
b  s  s     i  s  => (b: borrado, s: sustitución e i: inserción)
```

- Distancia: 5 (si cada operación cuesta 1)
- ¿De dónde sale el *?
  - Necesidad de alinear las cadenas de la forma más conveniente
  - Alineamiento correcto
  - Distancia: 5
- Algoritmo de Levenshtein

```
*  *  *  *  *  M  E  N  T  E
|  |  |  |  |  |  |  |  |  |
S  U  T  I  L  M  E  N  T  E
i  i  i  i  i
```

UNIVERSIDAD DE MURCIA

Unidad 4. Procesado del Lenguaje Natural

BAG-OF-WORDS Y TF-IDF

- Para la mayoría de las tareas de clasificación de texto, debemos representar cada documento por sus palabras
- Elementos básicos:
  - El corpus, formado por *N* documentos
  - El vocabulario, formado por *V* palabras
- La más básica es el bag-of-words (bolsa de palabras): Cantidad de veces que aparece una palabra en cada documento
- Term Frecuency (TF): ¿Cuántas veces aparece el término en el documento?
- Longitud del documento (|d|): ¿Cuál es la longitud del documento?
- Document Frequency (DF): ¿Cuántas veces aparece el término en algún documento?

UNIVERSIDAD DE MURCIA

- Ideas base para la representación de bag-of-words

$f(q = \text{“presidential campaign news”}, d)$

“Bag of Words”

$g(\text{“presidential”}, d)$ $g(\text{“campaign”}, d)$ $g(\text{“news”}, d)$

How many times does “presidential” occur in d?
Term frequency (TF): $c(\text{“presidential”}, d)$

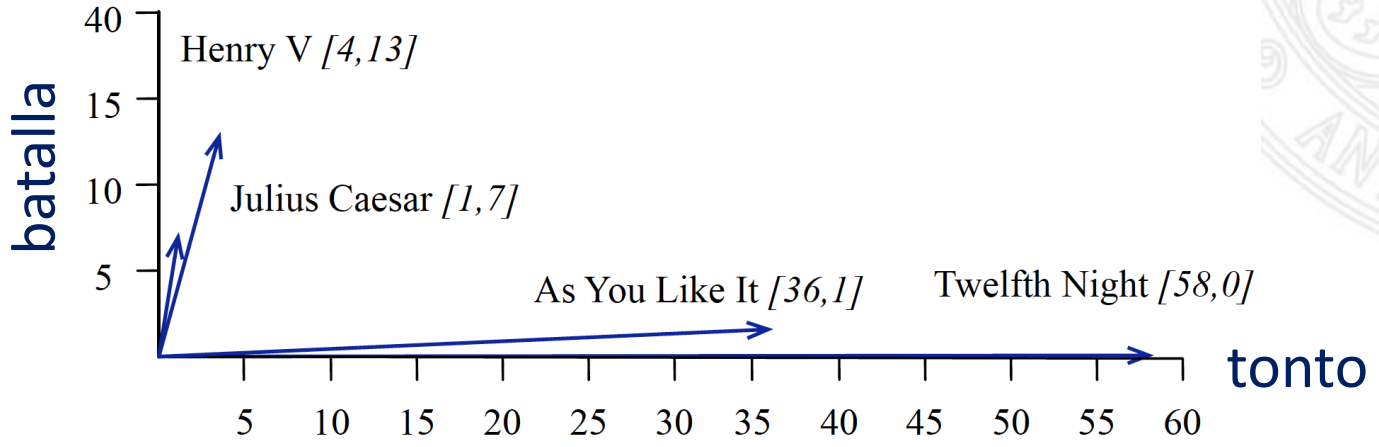
How long is d? **Document length:** $|d|$

How often do we see “presidential” in the entire collection?
Document frequency: DF(“presidential”)
P(“presidential”|collection)

Matriz de frecuencia término-documento

- Cada fila representa una palabra y la columna el documento
- Un documento puede ser representado por un vector contador
- El tamaño de los vectores tendría una dimensionalidad |V|
  - La matriz real sería de |V| filas y |D| columnas
- Usados originariamente para buscar documentos similares

	As you like it	Twelfth Night	Julius Caesar	Henry V
batalla	1	0	7	13
bueno	114	80	62	89
tonto	36	58	1	4
ingenio	20	15	2	3



TF-IDF: Term frequency (TF)

- Las frecuencias en crudo no son una buena aproximación
  - Hay sesgos y no son muy discriminativos (por ej, *bueno*)
  - Piensa en preposiciones o tamaño del documento
- Una mejora típica para esta situación es el uso los pesos TF-IDF (Term Frecuency – Inverse Document Frequency)
- Term frequency: La frecuencia de la palabra *p* en el documento *d*: $tf_{p,d} = count(p, d)$
- Frecuentemente se acortan las frecuencias con el $\log_{10}$ – Una palabra que aparezca 100 veces más no doble importancia
  - $tf_{p,d} = log_{10}(count(p, d) + 1)$
  - Una palabra que ocurre 0 veces tendría un $\log_{10}(1) = 0$

TF-IDF: Documento frequency (DF)

- El segundo factor es usado para dar más importancia a palabras que ocurren en pocos documentos
  - Esos términos son útiles para discriminar documentos
- Document frequency: El número de veces que una palabra *p* ocurre al menos una vez en un documento
  - No es lo mismo que collection frequency, que sería el número de veces que una palabra p ocurre en el Corpus entero

	Collection Frequency	Document Frequency
Romeo	113	1
acción	113	31

TF-IDF: Inverse document frequency (IDF)

- Se enfatiza aún más palabras como Romeo calculando el inverse document frequency
  - Se calcula: $\frac{N}{df_p}$ donde *N* es el número de documentos en la colección y df_p es el número de documentos en el que *p* aparece
  - Cuantas menos veces ocurra mayor es el peso
  - El peso más bajo sería 1, indicando que la palabra *p* aparece en todos los documentos
  - Debido al largo número de documentos también se suele usar el $\log_{10}$
- La definición final sería $idf_p = log_{10}(\frac{N}{df_p})$

df e idf en 37 obras

Palabra	df	idf
Romeo	1	1.57
ensalada	2	1.27
Falstaff	4	0.967
bosque	12	0.489
batalla	21	0.246
ingenio	34	0.037
tonto	36	0.012
bueno	37	0
dulce	37	0

TF-IDF: Todo junto

- Por lo tanto, el término tf-idf para una palabra $w_{p,d}$ para una palabra *p* en un documento *d* será: $w_{p,d} = tf_{t,d} * idf_t$
- En base a esto, podemos ver como se actualizan los términos que teníamos previamente
- Por ejemplo, el 0.049 de ingenio en As You Like It es el producto de tf = $\log_{10}(20+1)$ = 1.322 e idf = 0.037

	As you like it	Twelfth Night	Julius Caesar	Henry V
batalla	0.074	0	0.22	0.28
bueno	0	0	0	0
tonto	0.019	0.021	0.0036	0.0083
ingenio	0.049	0.044	0.018	0.022

UNIVERSIDAD DE MURCIA

Unidad 4. Procesado del Lenguaje Natural

N-GRAMAS

- Interesante saber la probabilidad de una sentencia (o como de común es) ya que tiene varias aplicaciones:
- Traducción automática
  - P("globulos rojos") > P("globulos colorados")
- Corrección ortográfica automática:
  - “El mercado está a diez minuts de mi casa.”
  - P(“El mercado está a diez minutos de mi casa”) > P(“El mercado está a diez minutas de mi casa”)
- Reconocimiento del habla
  - P(vientos fuertes) > P(bien, tos fuerte)
- Hay muchas otras aplicaciones

La probabilidad de una oración

- Calcular la probabilidad de una oración **W**, formada por palabras *w* (*words*): P(W) = P(w_1,w_2,w_3,w_4,w_5…w_n)
- Si fuera posible calcular dicha probabilidad, también podríamos calcular la probabilidad de la siguiente palabra, dada una oración incompleta: P(w_5 | w_1,w_2,w_3,w_4)
- A esto los llamamos **modelos de lenguaje** (language models)
- Hay que hacer cálculo de probabilidades. Por ejemplo, ¿cuál es la probabilidad de "El agua es transparente"?: P(el, agua, es, transparente)
- Regla de encadenamiento de probabilidades:
  - $P(A,B,C,D) = P(A) * P(B|A) * P(C|A,B) * P(D|A,B,C) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$

P(el, agua, es, tan, transparente) =
P(el)*P(agua|el)*P(es|el agua)*P(tan|el agua es)*P(transparente|el agua es tan

Intuición de los n-gramas

- $\prod_{k=1}^{n} P(w_k|w_{1:k-1})$ ¿Es viable estimar esta probabilidad en base a un número limitado de textos?
  - No del todo, porque el lenguaje es creativo y quizás haya frases que no hayan sucedido nunca
- La intuición del modelo *n*-grama es que estimaremos esa probabilidad sólo mirando las últimas *n* palabras, en vez de toda la historia
- El modelo más sencillo es el unigrama: $P(w_1, w_2, w_3 \dots w_n) \approx P(w_1) * P(w_2) * P(w_3) \dots P(w_n)$
- Si consideramos la dependencia con la palabra anterior tendremos el bigrama:$P(w_i|w_1, w_2, w_3 \dots w_{i-1}) \approx P(w_i|w_{i-1})$
- N-grama: Extenderíamos esta idea a las *n* palabras previa
  - Se hacen más complejos
  - En general no es un modelo suficiente por largas dependencias

- Podemos estimarlas mediante la estimación de máxima verosimilitud (MLE, lo mismo que hicimos con Naïve Bayes)
- La probabilidad de que la palabra w_i venga después de w_{i-1} vendrá dada por: $P(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)}{count(w_{i-1})}$
- Supongamos que este es nuestro corpus:

<s>En un plato de trigo</s>
<s>tres tristes tigres</s>
<s>comen trigo</s>

- Los símbolos <s> y </s> son caracteres metalingüísticos que indican comienzo y fin de oración respectivamente
  - Evaluar la probabilidad de estar al principio o al fin de sentencia

- Cálculo de las probabilidades:
  - P(en|<s>) = 1/3 = 0,33
  - P(un|en) = 1/1 = 1
  - P(plato|un) = 1/1 = 1
  - P(trigo|de) = 1/1 = 1
  - P(</s>|trigo) = 2/2 = 1
  - P(</s>|tigres) = 1/1 = 1
  - P(tres|<s>) = 1/3 = 0,33
  - P(en|<s>) = 1/3 = 0,33
  - P(de|plato) = 1/1 = 1
- ¿Cuál será P(<s>En un plato de trigo</s>)?
  - P(en|<s>) * P(un|en) * P(plato|un) * P(de|plato) * P(trigo|de) * P(</s>|trigo) = 0,33 * 1 * 1 * 1 * 1 * 1 = 0,33

Unidad 4. Procesado del Lenguaje Natural
Ejemplo real: Berkeley Restaurant Project

- Proyecto del Berkeley Restaurant Project. Sistema de diálogo que respondía preguntas de restaurantes en Berkley
- Resultados del cuenteo de bigramas para 8 palabras que tienen coherencia entre ellas
  - Si no son aún más escasas

	yo	quiero	para	comer	china	comida	almuerzo	gastar
yo	5	827	0	9	0	0	0	2
quiero	2	0	608	1	6	6	5	1
para	2	0	4	686	2	0	6	211
comer	0	0	2	0	16	2	42	0
china	1	0	0	0	0	82	1	0
comida	15	0	15	0	1	4	0	0
almuerzo	2	0	0	0	0	1	0	0
gastar	1	0	1	0	0	0	0	0

Ejemplo real: Berkeley Restaurant Project

- Probabilidades de los bigramas en función de los unigramas

yo	quiero	para	comer	china	comida	almuerzo	gastar
2533	927	2417	746	158	1093	341	278

	yo	quiero	para	comer	china	comida	almuerzo	gastar
yo	0.002	0.33	0	0.0036	0	0	0	0.00079
quiero	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
para	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
comer	0	0	0.0027	0	0.021	0.0027	0.0056	0
china	0.0063	0	0	0	0	0.52	0.0063	0
comida	0.014	0	0.014	0	0.00092	0.0037	0	0
almuerzo	0.0059	0	0	0	0	0.0029	0	0
gastar	0.0037	0	0.0036	0	0	0	0	0

- ¿P(<s> yo quiero comida china </s>) dado que P(yo|<s>) = 0.25 y P(</s>|china) = 0.68
- = P(yo|<s>) * P(quiero|yo) * P(comida|quiero) * P(china|comida) * P(</s>|china) = 0.25 * 0.33 * 0.0065 * 0.68

- Con un modelo de bigramas capturamos sobre todo hechos de naturaleza sintáctica
  - Por ejemplo, después de "*comer*" vendrá probablemente un nombre o adjetivo o que después de "*para*" vendrá un verbo
  - Otras culturales, como más comida china que inglesa
- Sólo hemos descrito bigramas con fines pedagógicos, pero los trigramas o los 4- y 5-gramas son más comunes cuando hay datos suficientes
- Debido a la multiplicación de probabilidades puede ser muy baja, se usan logaritmos para evitar underflow (desbordamiento por abajo)