Procesamiento de Lenguaje Natural

Eduardo Morales, Enrique Sucar

INAOE

Contenido

Introducción

Expresiones Regulares

N-Gramas

Part-of-Speech

Gramáticas Formales

Parseo Sintático

Parseo Estadístico

Semántica Léxica

Recuperación de Información

Traducción Automática

Procesamiento de Lenguaje Natural

La idea de dar a las computardoras la habilidad de procesar lenguaje natural es tan vieja como las computadoras. Ejemplos de sistemas:

1. Agentes conversacionales (e.g., HAL 9000) involucrando reconocimiento autómatico y síntesis automática
2. Traducción automática
3. Sistemas de pregunta-respuesta basadas en la Web/extracción de información, incluyendo inferencia, extracción de información y desambiguación de información
4. Otras áreas como, corrección de escritura, gramática, etc.

Procesamiento de Lenguaje Natural

Lo que distingue a procesamiento de lenguaje de otras áreas es el uso de *conocimiento del lenguaje*:

- El reconocimiento y síntesis de voz requiere de conocimiento acerca de la fonética y fonología de cómo se pronuncian las palabras
- El reconocer las variantes que tienen las palabras requiere de conocimientos de morfología
- Se requiere de un conocimiento de la estructura del lenguaje y cómo se agrupan las palabras o su sintáxis
- Para entender el significa de cada palabra se requiere de conocimiento del significado léxico y de un conjunto de palabras de semántica composicional.
- Para entender las intenciones del lenguaje se requiere de conocimiento de pragmática y diálogo.
- Finalmente para entender y asociar información dentro del lenguaje se requiere conocimiento del discurso

Áreas

- Fonética y fonología: Conocimiento acerca de los sonidos ligüísticos
- Morfología: Conocimiento acerca de la composición de las palabras
- Sintáxis: Relaciones estructurales entre las palabras
- Semática: Sentido
- Pragmática: Relaciones entre el significado de las metas e intenciones del hablante
- Discurso: Unidades lingüísticas mayores que una “unidad”

Ambigüedad

- La ambigüedad (múltiples alternativas) se da en la mayoría de las tareas de procesamiento de lenguaje
- Por ejemplo, en inglés: *I made her duck*
  1. Le cociné un pato
  2. Le cociné su pato
  3. Le hice un pato que ella posee
  4. Provoqué que se agachara rápido
  5. La convertí en pato
- Algunos problemas son que las palabras pueden tener varios sentidos (*duck* puede ser verbo o sustantivo, *make* puede significar hacer o cocinar), si se hubiera dicho (*made* suena también como *maid* y *I* suena también como *eye*)

Ambigüedad

Para resolver algunas ambigüedades se puede:

- *Part-of-speech tagging* (etiquetados de partes de la oración)
- *Word-sense disambiguation*
- *Lexical disambiguation*
- *Sintactic disambiguation*
- *Probabilistic parsing*
- *Speech-act interpretation*

Modelos

Mucho del conocimiento necesario para el procesamiento de lenguaje natural se puede capturar usando sólo algunos modelos

- Máquinas de estado: determinísticas, no determinísticas; Lenguajes regulares, Gramáticas libres de contexto, etc. Usados en fonología, morfología y sintaxis
- Lógica: Usada en análisis semático y en pragmática
- Modelos probabilistas: Los anteriores se pueden extender para incorporar incertidumbre. Modelos Markovianos, modelos ocultos de Markov (HMM), usados en reconocimiento de voz, análisis de diálogo, traducción automática, etc.
- Modelos de espacio de vectores: Usados en recuperación de información y en el significado de las palabras

Modelos y Algoritmos

Estos modelos usan un conjunto pequeño de algoritmos

- Búsqueda en espacio de estados: depth-first search, A*, ...
- Aprendizaje: árboles de decisión, SVM, mezcla de gaussianas, regresión logísitca, HMM, EM, NN, ...
- Técnicas de validación cruzada

Lenguaje y Máquinas Inteligentes

- Para muchos la habilidad de procesar lenguaje como los humanos es un signo claro de máquinas inteligentes
- La prueba de Turing: En escencia dice que es suficiente usar unicamente lenguaje para probar si una máquina se puede considerar que piensa
- Aunque en 1950 predijo que en 50 años se lograría, en 1966 se hizo un sistema, ELIZA, que engañó a varias personas
- Desde 1991 se establece el Loebner Prize
- Avances recientes en sistemas conversacionales (e.g., Siri) muestran a las personas tratando a las máquinas como personas

Ejemplos

Se han tenido avances recientes muy importantes gracias a Internet y las cantidades masivas de información:

- Reservaciones de trenes usando un sistema conversacional
- Control por medio de voz de sistemas en los vehículos (clima, música, sistema de navegación)
- Búsqueda de información en videos usando reconocimiento de voz
- Sistemas de traducción automática (Google translate, DeepL)
- Sistemas de calificación automática de pruebas de estudiantes
- Agentes virtuales interactivos que sirven como tutores
- Análisis automático de textos, opinión, preferencias, plagios, etc.

Historia

1) Conocimientos fundacionales: 1940s y 1950s

- Teoría de Autómatas: Turing (Máquina de Turing - 1936), McCulloch-Pitts (Red primitiva para lógica proposicional - 1943), Kleene (autómatas finitos y expresiones regulares - 1951), Chomsky (gramáticas libres de contexto - 1956)
- Modelos probabilistas y basados en teoría de la información: Shannon (modelos probabilistas para procesos markovianos y teoría de la información y entropía - 1948)
- También se dieron los inicios de reconocimiento de voz (reconocimiento de dígitos en Bell Labs - 1952)

Historia

2) Las dos áreas: 1957 - 1970

- La corriente simbólica: Por un lado siguiendo el trabajo de Chomsky y la teoría formal de lenguajes y por otro lado el área de Inteligencia Artificial
- La corriente probabilista: Enfoque Bayesiano para reconocimiento de texto
- También en ese periodo se creó un primer *corpus*

Historia

3) Cuatro paradígmas: 1970 - 1983

- La corriente probabilísta desarrolló los Modelos Ocultos de Markov (HMM)
- La corriente simbólica desarrollaron Prolog, las *Definitive Clause Grammars* y se trabajó en gramáticas funcionales y lexico-funcionales
- *Natural Languaje Understanding*: El sistema SHRDLU de Winograd (1971) con una gramática del inglés, Shank y colaboradores con la creación de los *scripts* (1977), Quinlan y el uso de redes semánticas (1968) y Filmore con los *case roles* (1968).
- Modelo del discurso: Inicios de estudios de las sub-estructuras del discurso, enfoque del discurso, resolución automática de referencias y sistemas de BDI (*belief-desire-intention*) usando *actos de habla*

Historia

4) Regreso del empirismo y modelos de estados finitos: 1983-1993

- Los modelos de estados finitos se volvieron a usar en fonología, morfología y sintáxis
- El empirismo creó el surgimiento de los modelos probabilistas en procesamiento de voz y lenguaje natural basados en datos y la creación de métricas de evaluación/comparación de algoritmos

Historia

5) Unificación del Área: 1994-1999

- Los modelos probabilistas y basados en datos se volvieron estándares en procesamiento de lenguaje natural
- Aumentó la capacidad de memoria y velocidad de las máquinas permitiendo poductos comerciales en reconicimiento de voz y corrección de escritura y gramática
- El surgimiento de Internet creó la necesidad de desarrollar algoritmos de recuperación y extracción de información

Historia

6) El ascenso del aprendizaje computacional: 2000-2008

- Grandes cantidades de material escrito y hablado se hicieron disponibles, con colecciones anotadas y evaluaciones más formales
- Concentración en aprendizaje computacional como herramienta
- Cómputo de alto rendimiento
- Modelos estadísticos no supervisados usados, por ejemplo, para traducción automática

Expresiones Regulares y Autómatas

- Expresiones regulares (Kleene 1956) se usan en “grep” de Unix, en Emacs, ... basadas en cadenas.
- Una expresión regular en una notación algebraica para caracterizar un conjunto de cadenas
- Se pueden usar para buscar patrones en textos.

Ejemplos

- /foo/
- /[Ff]oo/ ó /[1234567890]/ ó /[A-Z]/ ó /[∧A]/ (negación)
- *: Kleene (cero ó más)
- +: al menos 1
- /[0-9][0-9]*/ ó /[0-9]+/
- ...

Substituciones

- s/a/b/
- Ejemplo, ELIZA:
  - s/.* YOU ARE (depressed/sad) .*/I AM SORRY TO HEAR YOU ARE \1 /
  - s/.* YOU ARE (depressed/sad) .*/WHY DO YOU THINK YOU ARE \1 /
  - s/.* all .*/IN WHAT WAY/
  - s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/

Autómatas Finitos

- Las expresiones regulares son una forma de describir Autómatas Finitos (o de estados finitos).
- Cualquier expresión regular se puede representar por un autómata finito (y al revés).
- Representan (junto con las gramáticas regulares) lenguajes regulares
- También los podemos expresar con tablas de transición de estados

Definición Formal de un Autómata

Un AF se representa como una 5-tupla: $A = (Q, \Sigma, \delta, q_0, F)$.
Donde:

1. Q: Un conjunto finito de *estados*.
2. Σ: Un conjunto finito de *símbolos de entrada* (alfabeto)
3. δ: Función de transición (e.g., $\delta(q, a) = p$)
4. q_0: El estado *inicial/de comienzo*.
5. F: cero o más *estados finales/de aceptación*.

Ejemplo

Un Autómata A que acepta $L = \{x01y | x \wedge y \in \{0,1\}^*\}$

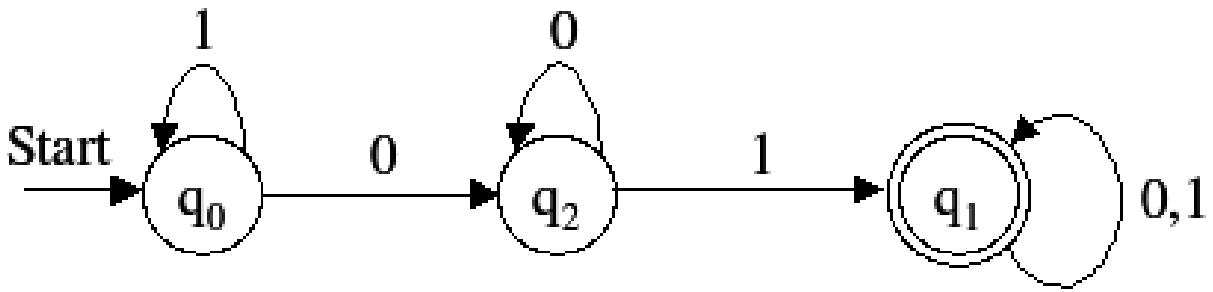
- El DFA acepta cadenas que tienen 01 en alguna parte de la cadena
- El lenguaje del DFA es el conjunto de cadenas que acepta $\{w|w$ tiene la forma "$x01y$" para algunas cadenas x y y que consisten sólo de 0's y 1's $\}$.

Tabla de Transiciones

El autómata anterior puede ser representado con una tabla de transiciones, definido como $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, q_1)$, de la siguiente forma:

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

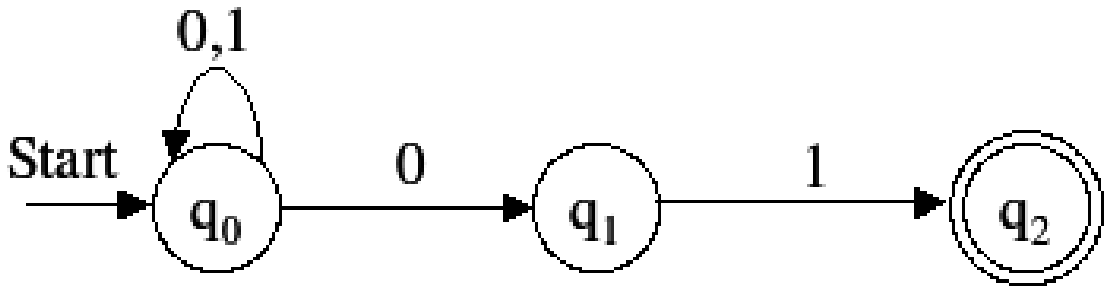
Diagrama de Transiciones



- Cada estado tiene un nodo asociado
- Cada transición de estados tiene un arco asociado etiquetado con el/los símbolos correspondientes
- Existe una etiqueta de inicio para el estado inicial y un doble círculo asociado a los estados finales

Autómata Finito No-Determinístico

Un autómata finito es no-determinístico cuando se permite que el AF tenga 0 o más estados siguientes para cada par estado-entrada:



Autómata Finito No-Determinístico

- Un NFA es una herramienta importante para diseñar procesadores de cadenas, e.g., *grep*, analizadores léxicos, etc. Es fácil diseñar NFAs que encuentren secuencias de palabras en texto.
- **NFA**: Formalmente, un NFA es una quíntupla $A = (Q, \Sigma, \delta, q_0, F)$, donde todo es un DFA, pero $\delta(q, a)$ nos regresa un conjunto de estados en lugar de un solo estado. De hecho puede ser vacío, tener un solo estado o tener más estados.
- Un NFA acepta, al igual que un DFA, lenguajes regulares

N-Gramas

- Un aspecto muy importante dentro del análisis del lenguaje es poder predecir las palabras
- Los N-gramas permiten predecir la siguiente palabra dados las $N-1$ palabras anteriores.
- Un 2-grama, conocido como bigrama, es una secuencia de dos palabras, e.g., si gracias, por favor, etc., y un 3-grama, conocido como trigrama, es de tres, e.g., a la derecha, cómo te llamas, etc.
- Estos modelos de secuencias de palabras también se conocen como modelos del lenguaje (LM).

N-Gramas

- Se pueden asignar probabilidades condicionales a secuencias de palabras, pero también se puede usar para asignar probabilidades a oraciones completas.
- Esto es muy útil en reconocimiento de voz, se puede usar en reconocimiento de escritura, en traducción automática y correción de escritura.
- También es útil en otras tareas de NLP como etiqueta de partes de la oración, generación de lenguaje natural, identificación de autoría, extracción de emociones, etc.
- Las estadísticas se hace sobre un corpus de texto o voz. Lo importante es, qué contar y cómo contar.

Consideraciones

- Se consideran puntuaciones o no? Las puntuaciones nos sirven para separar palabras.
- Que hay de *perro* y *perros* o todas las posibles declinaciones de un verbo? Contamos todas las palabras distintas o todas las raices distintas?
- Con N-gramas queremos calcular: $P(w|h)$, donde w se refiere a una palabra y h a un conjunto de palabras previas o su historia.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

dado $P(B) > 0$

N-Gramas

- Si queremos calcular la probabilidad de que: *tina* aparezca después de *anita lava la*, sería:

$$P(\text{tina} \mid \text{anita lava la}) = \frac{C(\text{anita lava la tina})}{C(\text{anita lava la})}$$

  donde C significa, contar en cuantos textos de nuestro corpus aparece esa secuencia de palabras.
- En general, estimar la probabilidad de un secuencia de palabras se reduce a contar el número de veces que aparace la secuencia entre el número total de secuencias de este tamaño.
- Esto es costoso, aplicar la regla de la cadena no ayuda mucho, y lo que generalmente se hace es suponer una breve historia.

N-Gramas

- Un bigrama aproxima:

$$P(\text{tina} \mid \text{anita lava la}) \approx P(\text{tina} \mid \text{la})$$

(suposición de Markov).

- Para estimar la probabilidad del bigrama y dada la palabra x, podemos calcular el número de bigramas $C(x, y)$ y normalizarlo por la suma de todos los bigramas de dos palabras que empiezan con x.

$$P(y \mid x) = \frac{C(x, y)}{\sum_w C(x, w)} = \frac{C(x, y)}{C(x)}$$

- Dado el cálculo de bigramas, podemos aproximar el cálculo de una oración multiplicando todos sus bigramas.
- Algo parecido se hace con trigramas: $P(\text{tina} \mid \text{lava la})$

N-Gramas

- Las estadísticas obviamente dependen del corpus del cual fueron obtenidas.
- En general, los N-gramas modelan mejor el corpus de entrenamiento al aumentar el valor de N.
- Pero con N grande tenemos, en algunos casos, muy pocas ocurrencias.
- Por ejemplo, si queremos contruir una oración con cuatrigramas, empezamos con las 3 palabras de inicio más comunes, seguida de la palabra más común dada las 3 anteriores, y así sucesivamente.

N-Gramas

- En algunos casos, y coforme aumentamos el tamaño de N es posible que no existan esas combinaciones de palabras.
- Para evaluar un modelo del lenguaje, podemos tener, por ejemplo, un reconocedor de voz que usa un modelo basado en bigramas contra uno que usa trigramas.
- Esto es costoso experimentalmente, por lo que muchas veces se usa una medida que es independiente de la aplicación, e.g., *perplexity* (perplejidad)

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

N-Gramas

- Con la regla de la cadena sería:

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

- Para bigramas sería:

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

- Esto se prueba en un conjunto de textos de prueba y se prefiere el de menor valor. También se introducen separados de oraciones.

Suavizado

- El problema con lo que hemos visto es que si alguna palabra no ocurre en el corpus, su probabilidad se vuelve cero y al estar multiplicando todo se hace cero.
- Para ésto, se han propuesto varias formas de "suavizado".
- Estimador Laplaciano:

$$P(w_i) = \frac{c_i + 1}{N + V}$$

Donde c_i es cuantas veces aparece la palabra w_i, N es el número de palabras en el corpus y V en el número total de palabras distintas.

Suavizado

- Good-Turing: Usar información de las palabras (eventos) que ocurren una sola vez para estimar los que no ocurren

$$P_{GT} = \frac{N_1}{N}$$

Donde N_1 es el número de bi-gramas que aparecen 1 vez y N es el número de N-gramas diferentes
En general, estimar las cosas que pasan c veces usando las que ocurren $c+1$ veces.

$$c* = (c+1)\frac{N_{c+1}}{N_c}$$

- Se han hecho varios extensiones usando información de N-gramas de orden menor para estimar los de orden superior

Part-of-Speech Tagging

- *Part-of-Speech* (POS) se ha hecho desde hace mucho tiempo (e.g., Dionysius Thrax de Alejandría 100 A.C. - nombre, pronombre, verbo, preposición, adverbio, conjunción, participio y artículo)
- La idea es asignar categorías o etiquetas a las palabras
- Puede ser genérico como verbo o sustantivo o más particular, como pronombre posesivo y pronombre personal

Part-of-Speech Tagging

- Se pueden dividir en clases cerradas (preposiciones, conjunciones, determinantes, pronombres) o abiertas (sustantivos, verbos, adjetivos, adverbios)
- Existe mucho desarrollo y herramientas (principalmente para el inglés) que anotan automáticamente texto
- Ejemplo: The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Part-of-Speech: Penn Treebank

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Part-of-Speech Tagging

- Muchas de las etiquetas son difíciles de asignar, inclusive para las personas
- Existen palabras que pueden tener más de una etiqueta (ambigüedad)
- Existen dos clases de algoritmos:
  1. Basados en reglas: (i) Generan una gran cantidad de etiquetas y (ii) aplican restricciones. E.g., EngCG
  2. Basados en probabilidad: Calculan la secuencia de etiquetas más probable dada una secuencia de palabras, haciendo simplificaciones. E.g., HMM tagger.

$$\hat{t}_1^n = argmax_{t_1^n} P(w_1^n|t_1^n)p(t_1^n)$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Part-of-Speech Tagging

Problemas:

- Ambigüedad: Cómo manejar etiquetas múltiples
- Palabras desconocidas: Asignarle todas y que las palabras vecinas ayuden a quitar la ambigüedad
- POS en otros idiomas: Existen lenguas como el turco que tiene un vocabulario mucho más extenso que el inglés y que las etiquetas se tienen que aumentar con otros aspectos, como género o número
- A veces, como con los ensambles de clasificadores, se corren varios programas y se combinan sus resultados

Gramáticas Formales

- Las gramáticas existen desde hace mucho tiempo y están relacionadas con la sintaxis
- La sintaxis se refiere a la forma en que las palabras están organizadas
- Existen 3 conceptos importantes que se capturan con las gramáticas:
  1. Constituyentes: Grupos de palabras que se comportan como una unidad, e.g., “a las 4” no se puede separar
  2. Relaciones gramaticales: Relaciones entre categorías, e.g., “sujeto” seguido de “predicado”
  3. Sub-categorización y dependencias: Relaciones entre palabras y frases, e.g., ciertas palabras no pueden asociarse con ciertas frases

Gramáticas Formales

- Una gramática libre de contexto (CFG) consiste de un conjunto de *reglas* o *producciones* que expresan la forma en que los símbolos de un lenguaje se pueden agrupar y un *lexicón* de palabras y símbolos
- Por ejemplo:
  $S \rightarrow NP\ VP$
  $NP \rightarrow Det\ Noun$
  $NP \rightarrow ProperNoun$
  $VP \rightarrow Verb\ NP$
  $VP \rightarrow Verb\ NP\ PP$
  $VP \rightarrow Verb\ PP$
  $PP \rightarrow Preposition\ NP$
- Una CFG se puede usar para generar oraciones o para generar una estructura a una oración (árbol de parseo)

Definición formal de CFGs

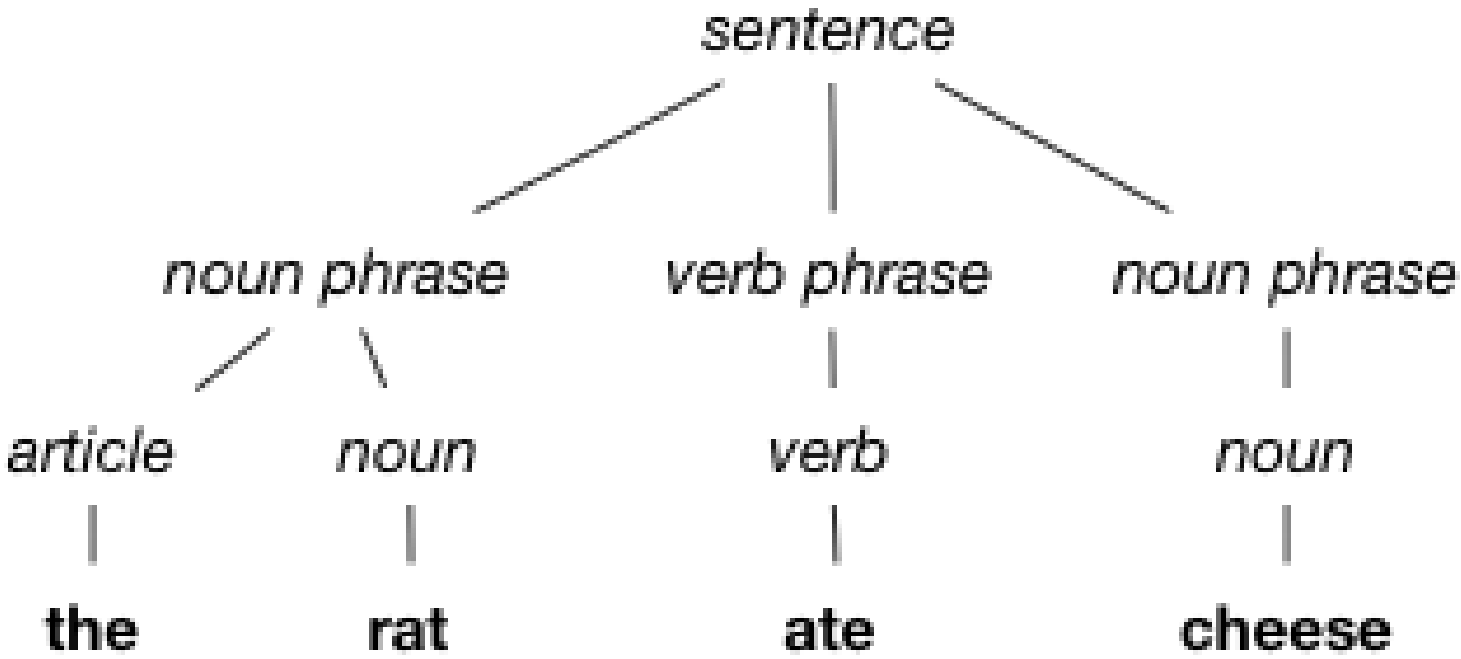
Una grámatica libre de contexto se define con $G = (V, T, P, S)$ donde:

- V es un conjunto de *variables*
- T es un conjunto de *terminales*
- P es un conjunto finito de *producciones* de la forma $A \rightarrow \alpha$, donde A es una variable y $\alpha \in (V \cup T)^*$
- S es una variable designada llamada el *símbolo inicial*

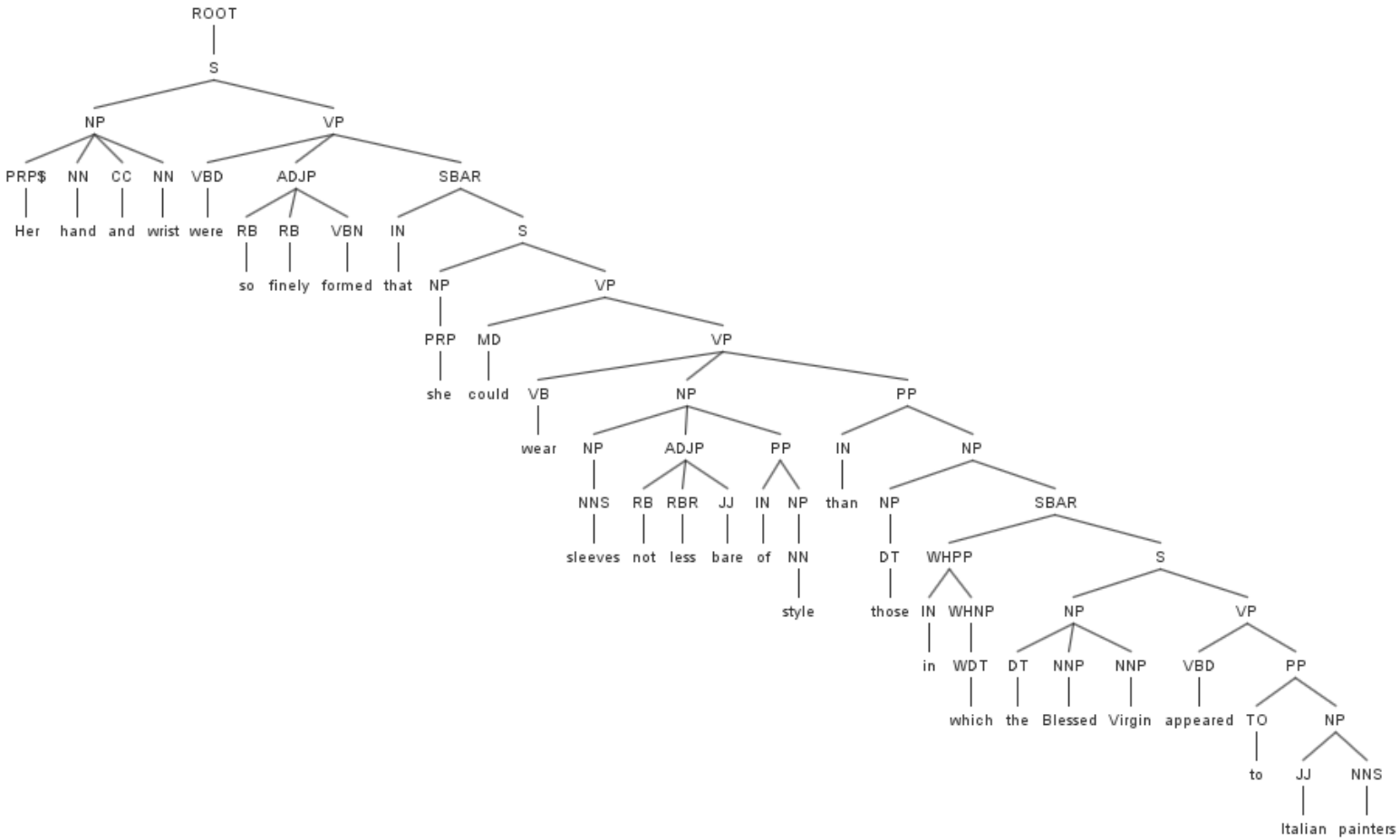
Derivaciones

- Podemos usar la gramática para reconocer si una cadena de símbolos pertenece al lenguaje definido por la gramática
- Podemos usar la gramática para generar cadenas que pertenecen a la gramática
- Sea $G = (V, T, P, S)$ una CFG, $A \in V, \{\alpha, \beta\} \subset (V \cup T)^*$ y $A \rightarrow \gamma \in P$.
- Escribimos: $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$ o si se sobre-entiende G: $\alpha A \beta \Rightarrow \alpha\gamma\beta$ y decimos que $\alpha A \beta$ *deriva* $\alpha\gamma\beta$
- Definimos $\overset{*}{\Rightarrow}$ como la cerradura reflexiva y transitiva de $\Rightarrow$
- Si $G(V, T, P, S)$ es una CFG, entonces el lenguaje de G es: $L(G) = \{w \in T^* : S \overset{*}{\Rightarrow}_G w\}$

Árboles de Parseo



Árboles de Parseo



Árboles de Parseo

Learning to Parse: A Taste

- Penn Treebank project (about 1M words)

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN ))))
```

(b)

Figure 12.7 Parsed sentences from the LDC Treebank3 version of the Brown (a) and ATIS (b) corpora.

Gramáticas Formales

- Se han realizado varias extensiones a las CFGs, e.g., las *Definite Clause Grammars* o DCGs, se definieron cuando se creó el sistema Prolog. E.g.:
  oración $--> $ sujeto(Num), predicado(Num).
  sujeto(N) $--> $ art(Num,Gen), sust(Num,Gen).
  predicado(Num) $--> $ verbo(Num), sujeto(_).
  art(sing,fem) $--> $ [la].
  ...
  verbo(sing) $--> $ [come].
  verbo(plural) $--> $ [comen].
  sust(sing,masc) $--> $ [hombre].
  sust(sing,fem) $--> $ [manzana].
  ...

Parseo

Existen dos estrategias básicas:

1. Top-down o goal-directed: Construye un ábol de parseo empezando por el símbolo inicial (S) o nodo raíz. Se supone que la cadena se puede derivar a partir del símbolo inicial y se exploran todos los árboles (en principio) en paralelo. No explora sub-árboles que no van a funcionar.
2. Bottom-up o data-directed: Empieza con las palabras y construye árboles a partir de ellas usando reglas de la gramática. No construye árboles inconsistentes con la entrada.

Árboles de Parseo

1. Initial state

Stack	Remaining Text
	the dog saw a man in the park

2. After one shift

Stack	Remaining Text
the	dog saw a man in the park

3. After reduce shift reduce

Stack	Remaining Text
Det N the dog	saw a man in the park

4. After recognizing the second NP

Stack	Remaining Text
NP V NP in Det N saw Det N the dog a man	the park

5. After building a complex NP

Stack	Remaining Text
NP V NP Det N saw NP PP the dog Det N P NP a man in Det N the park	

6. Built a complete parse tree

Stack	Remaining Text
S NP VP Det N V NP the dog saw NP PP Det N P NP a man in Det N the park	

Ambigüedad Sintáctica en el Parseo

- *Attachment ambiguity*: “Anoche maté a un elefante en pijamas” o “Vi a un conejo en la montaña con binoculares”
- *Coordination ambiguity*: “Los hombres y mujeres de la tercera edad” o “se escucho en la radio y la televisión a nivel nacional”
- El problema es que pueden existir muchas posibles interpretaciones (parseos)
- Aunque la oración no tenga problemas, las palabras pueden tener más de una interpretación

Parseo

- Se hacen búsquedas con *backctracking*, con programación dinámica (guardando los subárboles para no hacer *backtracking* sobre lo mismo más de una vez)
- Existen muchas variantes de algoritmos:
  - Parseo CKY: La gramática tiene que estar en CNF (reglas tipo: $A \rightarrow BC$ o $A \rightarrow w$), sigue una estrategia *bottom-up* y se basa en llenar una tabla triangular de izquierda a derecha
  - Algoritmo Earley: Usa programación dinámica en una estrategia *top-down*, hace una sola pasada de izquierda a derecha llenando un arreglo representando los árboles parciales generados hasta el momento.
  - Chart Parsing: Utiliza una agenda, la cual se ordena de acuerdo a una cierta política, que permite más flexibilidad.

Parseo

- También se pueden hacer parseos parciales
- Realizar *chunking* (crear pedazos, como frases nominales, etc.)
- Usar aprendizaje con texto anotado

Parseo Probabilista

- Las técnicas anteriores pueden representan las diferentes interpretaciones, pero no resuelven la ambigüedad
- Un algoritmo de parseo probabilista encuentra la probabilidad de ocurrencia de cada interpretación y regresa la más probable
- Son útiles para entender lenguaje hablado, prediciendo las posibles palabras o resolviendo posibles interpretaciones
- La gramática más comunmente usada en la gramática libre de contexto probabilista (PCFG)

PCFG

- Igual que una CFG, pero las reglas (e.g., $A \rightarrow \beta$) tienen asociado un número (p) que representa la probabilidad que el símbolo no terminal (A) se expanda a su secuencia (β)
- Se puede expresar como:

$$P(A \rightarrow \beta) \text{ ó } P(A \rightarrow \beta \mid A) \text{ ó } P(RHS \mid LHS)$$

- Por lo que:

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

Parseo Probabilista

- La probabilidad de un árbol de parseo (T) se define como el producto de las n reglas que se usaron para derivarlo
- También se puede usar para encontrar la probabilidad de una frase sumando las probabilidades de todos sus árboles
- Se pueden usar para encontrar la probabilidad de una frase o una(s) palabra(s), e.g., $P(w_i \mid w_1, w_2, \ldots, w_{i-1})$

Parseo Probabilista

- Muchos de los algoritmos de parseo actuales se basan en parseo probabilista, e.g., CKY probabilista
- Las probabilidades se pueden obtener de algún *corpus*:

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\mid \alpha \rightarrow \beta \mid}{\sum_{\gamma} \mid \alpha \rightarrow \gamma \mid} = \frac{\mid \alpha \rightarrow \beta \mid}{\mid \alpha \mid}$$

- También se han hecho desarrollos para tratar de aprender las gramáticas

Parseo Probabilista

- Algunos de los problemas de las PCFG es su suposición de independencia en las partes (por eso se multiplican las probabilidades)
- Para resolverlo muchas veces se aumentan las PCFG con dependencias léxicas y estructurales
- Uno de los más usadas es el de Collins (1999)

Semántica Léxica

- Se trata de entender el significado de las palabras (lexemas)
- Lexema: Se representa por su lema (*lemma*). E.g., en verbos se representa por su infinitivo
- Lexicón: Conjunto de lexemas
- Para entontrar los lemas se hace un proceso de lematización (*lemmatization*), el cual no es determinísta (e.g., *banco*)

Semántica Léxica

- Sinonímia: Dos palabras que se escriben diferentes pero tienen el mismo (o casi) significado. E.g., coche y automovil
- Antonímia: Dos palabras con significados opuestos. E.g., alto y bajo.
- Hiponímia: Una palabra es más específica que otra. E.g., mango y fruta.
- Hiperímia: Una palabra es más general que otra. E.g., fruta y mango.
- Uno puede pensar en construir taxonomías y ontologías

WordNet

- La fuente de inglés del sentido de las palabras más utilizada
- Lo pueden consultar o bajar localmente
- Representa los *synset*s - conjunto de sinónimos o sentidos de las palabras

Dan Jurafsky

WordNet Noun Relations

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	$breakfast^1 \rightarrow meal^1$
Hyponym	Subordinate	From concepts to subtypes	$meal^1 \rightarrow lunch^1$
Instance Hypernym	Instance	From instances to their concepts	$Austen^1 \rightarrow author^1$
Instance Hyponym	Has-Instance	From concepts to concept instances	$composer^1 \rightarrow Bach^1$
Member Meronym	Has-Member	From groups to their members	$faculty^2 \rightarrow professor^1$
Member Holonym	Member-Of	From members to their groups	$copilot^1 \rightarrow crew^1$
Part Meronym	Has-Part	From wholes to parts	$table^2 \rightarrow leg^3$
Part Holonym	Part-Of	From parts to wholes	$course^7 \rightarrow meal^1$
Substance Meronym		From substances to their subparts	$water^1 \rightarrow oxygen^1$
Substance Holonym		From parts of substances to wholes	$gin^1 \rightarrow martini^1$
Antonym		Semantic opposition between lemmas	$leader^1 \Longleftrightarrow follower^1$
Derivationally Related Form		Lemmas w/same morphological root	$destruction^1 \Longleftrightarrow destroy^1$

WordNet

Dan Jurafsky

WordNet VerbRelations

Relation	Definition	Example
Hypernym	From events to superordinate events	$fly^9 \rightarrow travel^5$
Troponym	From events to subordinate event (often via specific manner)	$walk^1 \rightarrow stroll^1$
Entails	From verbs (events) to the verbs (events) they entail	$snore^1 \rightarrow sleep^1$
Antonym	Semantic opposition between lemmas	$increase^1 \Longleftrightarrow decrease^1$
Derivationally Related Form	Lemmas with same morphological root	$destroy^1 \Longleftrightarrow destruction^1$

Papel Temático

- El *thematic role* se refiere al papel que juegan dentro de la oración las palabras, por ejemplo, *agente* (el que causa un evento), *tema* (la parte afectada por el evento), *experimentador* (de una acción), *resultado, beneficiario, instrumento*, etc.
- Ejemplo: Juan (agente) rompió la ventana (tema) con un ladrillo (instrumento)
- Los verbos permiten que los roles temáticos puedan ocurrir en diferentes posiciones sintácticas (e.g., el ladrillo rompió la ventana)
- Aunque funcionan en general, existen muchos casos en donde no es posible definir claramente los roles

Semántica Léxica Computacional

- El encontrar el sentido correcto de una palabra se conoce como *Word Sense Disambiguation* (WSD) o desambiguación del sentido de palabra
- Los algoritmos toman una palabra con sus diferentes significados en un contexto particular y regresan el significado adecuado
- Para esto se han usado clasificadores (entrenan con palabras con diferentes significados en diferentes contextos).
- Se tienen que definir los atributos a usar (e.g., N palabras antes y después, con POS tagging, lematización o *stemming* - la raíz de la palabra)

Enfoques: Colocación y BoW

1. Colocación: Se genera un vector con relaciones de la posición de las palabras con respecto a la palabra objetivo junto con sus POS. E.g., "An electric guitar and **bass** player stand off to one side ...".
   {guitar, NN, and, CC, player, NN, stand, VB}
2. Bolsa de palabras (BoW): Se selecciona un conjunto de las palabras vecinas más comunes dentro de un corpus (el orden de aparición en la oración no importa).
   E.g., las 12 palabras más comunes que son vecinas a *bass* en un corpus son: [fishing, big, sound, player, fly, rod, pond, double, runs, playing, guitar, band]
   Por lo que la palabra *bass* dentro del ejemplo anterior se representaría como: [0,0,0,1,0,0,0,0,0,0,1,0]

Semántica Léxica Computacional

- Dados conjuntos de bolsas de palabras con diferentes significados, se pueden entrenar un clasificador para decidir el significado correcto
- Una opción es usar un naïve Bayes (suponer independencia de las palabras/atributos dado el sentido/clase)

$$\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j \mid s)$$

- Tambien se pueden usar *Decision Lists* y árboles de decisión

El Algoritmo Lesk

- Uno de los métodos más comunes es utilizar información adicional de diccionarios o thesaurus para resolver la ambigüedad
- La idea es muy simple:
  - Obten las palabras usadas en la oración que contiene la palabra ambígua
  - Obten las palabras que existen en las definiciones de diccionario de la palabra ambígua
  - Regresa el sentido que tiene más palabras comunes con las de la oración
- Si no se tienen suficientes palabras, se puede expandir el conjunto de palabras y también pesarlas
- Para evitar la necesidad de muchos datos etiquetados se puede usar aprendizaje semi-supervisado

Similitud de palabras

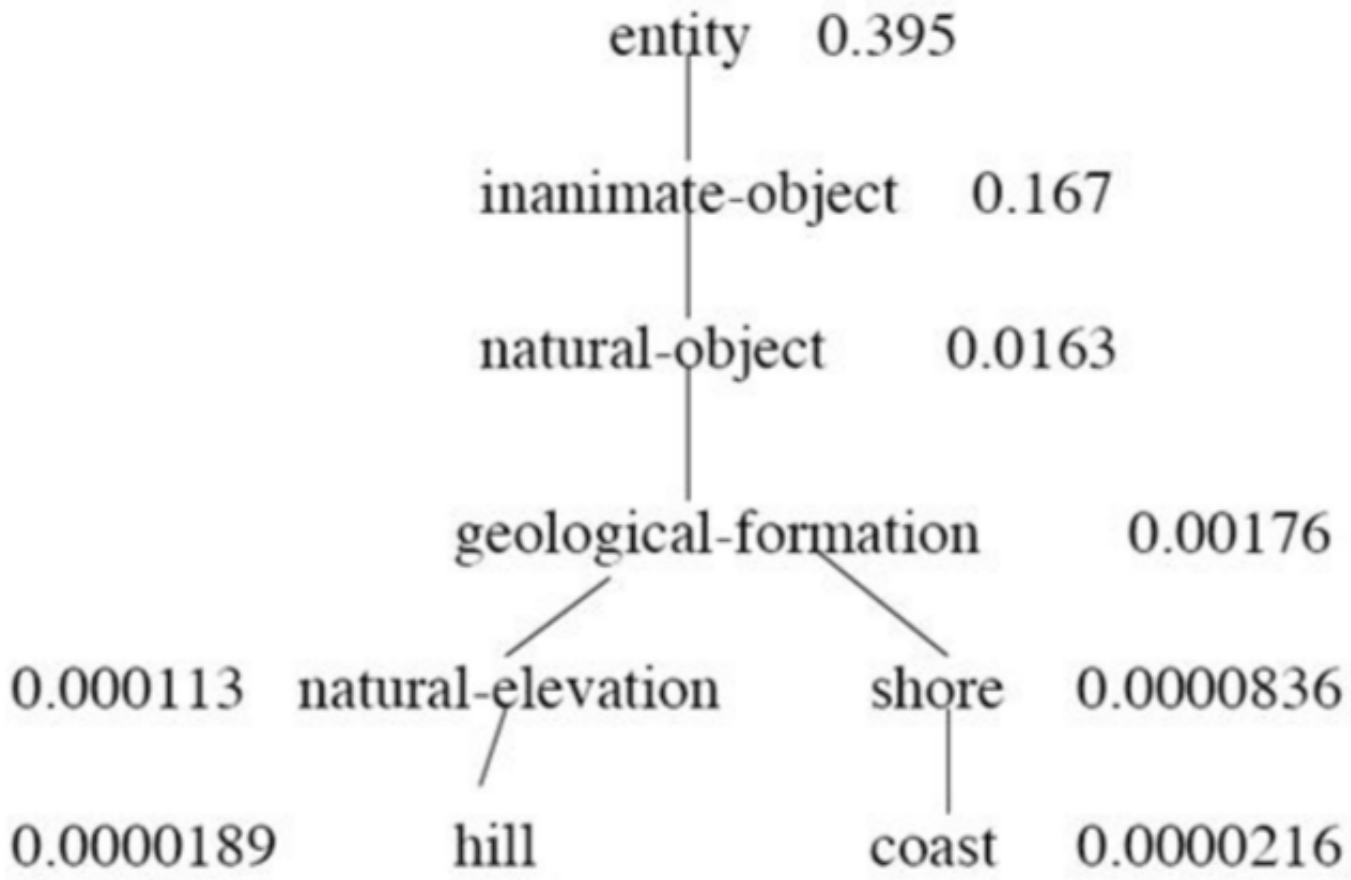
- Encontrar la similitud entre palabras es útil para recuperación de información, traducción automática, generación de resúmenes, etc.
- Algunos algoritmos se basan en thesaurus y encuentran distancias siguiendo relaciones tipo *is-a*, pero sólo funciona o entre sustantivos o entre verbos.

$$sim_{camino}(c_1, c_2) = -log \text{ long-camino}(c_1, c_2)$$

- También se puede buscar las relaciones de todo tipo entre dos palabras
- O buscar la máxima similaridad entre palabras similares de dos palabras

Semántica Léxica Computacional

Tomando en cuenta diferente peso dependiendo en qué tan probable es la clase:



Métodos Distribucionales

- El sentido de una palabra está relacionado con la distribución de las palabras que la rodean
- Se puede encontrar una matriz de co-ocurrencia de palabras. E.g.:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Word2Vec

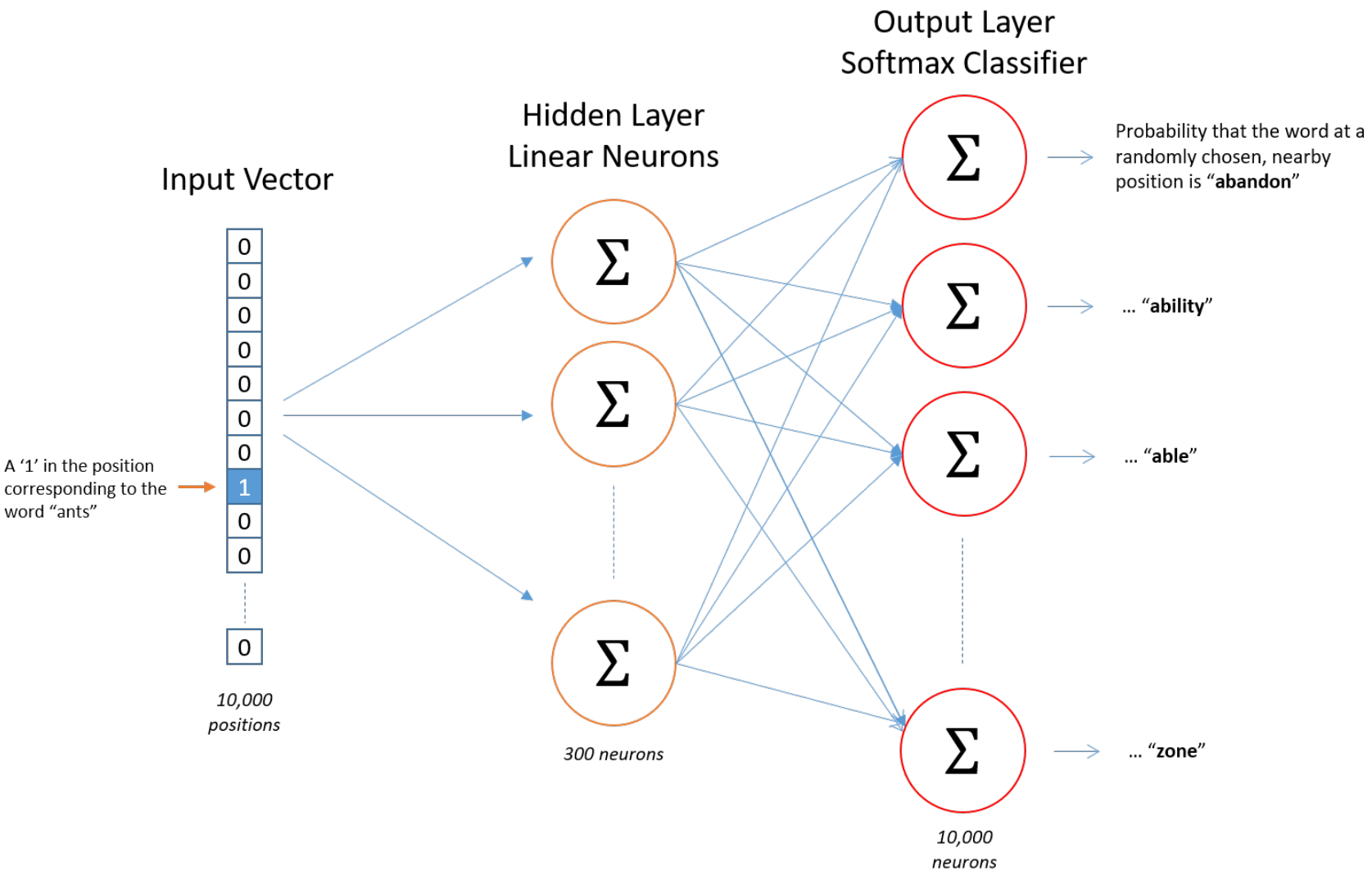
- Posiblemente el algoritmo que ha dominado más el encontrar palabras con semántica parecida
- La idea es usar un truco que se hace en otras áreas de ML: Aprender una red para una cosa, pero lo que te interesa es aprender los pesos intermedios que son los que usas para otro cosa, e.g., Auto-Encoder
- Entrenar una red para que dada una palabra (en una oración) nos diga, para cada palabra en el vocabulario, la probabilidad de que sea una palabra "cercana" (de acuerdo al tamaño de una ventana)
- Entrenamos la red dándole pares de palabras (fuente y cercana), como no le podemos dar las palabras, generamos vectores de nuestro vocabulario. E.g., para 10 mil palabras las entradas son vectores con 9,999 ceros y un 1 en la posición que corresponde a la palabra.

Word2Vec

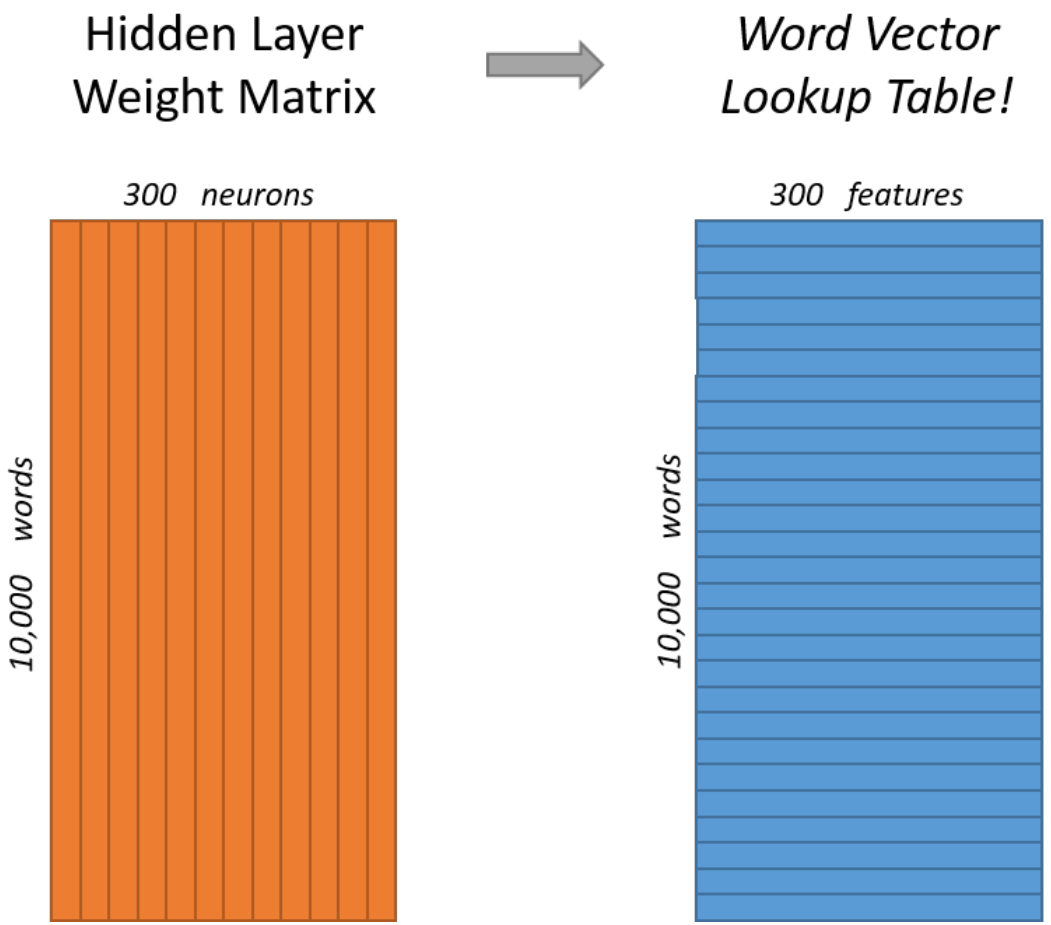
- La salida es igualmente un vector de 10 mil componentes representando la probabilidad de que cada palabra en el vocabulario sea vecina
- En medio se tiene una ventana oculta de N unidades, e.g., 300 (lo que usó Google) y se aprenden los pesos (una matriz de 10 mil x 300) que son lo que finalmente vamos a usar (desechamos el resto)
- Se puede ver como una *lookup table*

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Word2Vec

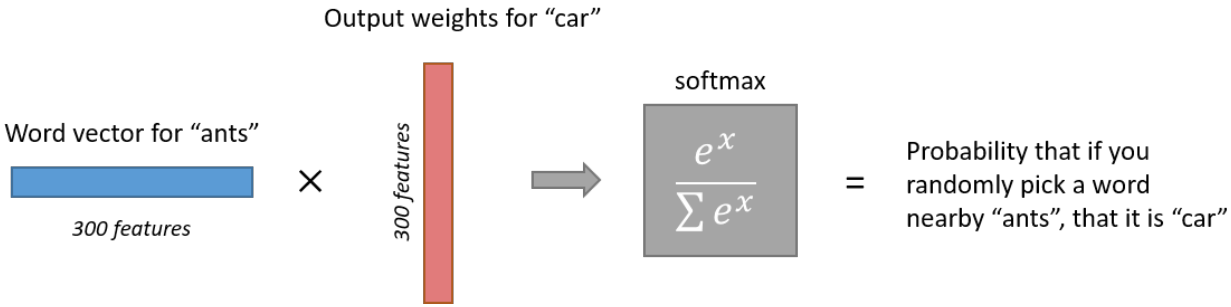


Word2Vec



Word2Vec

- La capa de salida es un clasificador de regresión softmax, donde cada salida de la red regresa un número entre 0 y 1 y la suma de todos es 1



- Dado que el número de pesos es gigantesco ($300 \times 10,000 \times 2$) se hacen varios "trucos" para poder entrenar la red:
  - Considerar conjuntos de palabras comunes como una sola
  - Realizar submuestreo de palabras frecuentes
  - Usar *negative sampling* que actualiza con cada ejemplo un conjunto pequeño de pesos

Word2Vec

- Al final a cada palabra se le asocia un vector de magnitud 300 y se pueden comparar si dos palabras tienen un significado parecido por qué tanto se parecen sus vectores
- E.g., Distancia Manhattan, Euclideana, Jaccard, Dice, KL, ...
- Coseno (la que más se usa por no depender del tamaño del vector):

$$coseno(\vec{c}, \vec{w}) = \frac{\vec{c} \cdot \vec{w}}{\mid \vec{c} \mid\mid \vec{w} \mid} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

Recuperación de Información

- *Information Retrieval* (IR) involucra la recuperación de documentos de todo tipo
- Muchos sistemas modelan el problema en un espacio de vectores, representando las palabras
- Se pueden modelar como vectores de 1's y 0's (si están o no están) o mejor aún por su frecuencia de apariciones en el documento
- Se puede usar una distancia coseno entre estos vectores de documentos para encontrar similaridad

Recuperación de Información

- Lo más utilizado es tener los términos pesados dándole más peso a los términos más frecuentes (TF) pero también considerando los más discriminativos (IDF)
- *Inverse document frequency* (IDF):

$$idf_i = log\left(\frac{N}{n_i}\right)$$

donde: N es el número total de documentos y n_i es el número de documentos en donde aparece la palabra i

- Combinándo estos dos factores los pesos que se usan para la palabra i en el vector del documento j son TF-IDF: $w_{i,j} = tf_{i,j} \times idf_i$
- Se prefieren (tienen más peso) términos frecuentes en el documento, pero raros en la colección

Mejoras a la Recuperación de Información

- Dado que existen palabras con varios sentidos, se pueden presentar documentos al usuario y que el seleccione los más relevantes (*relevance feedback*)
- También se pueden añadir términos que sean sinónimos para mejorar la recuperación (*query expansion*)
- Existen mucha literatura sobre sistemas que responden a preguntas concretas

Commercial systems: mainly factoid questions

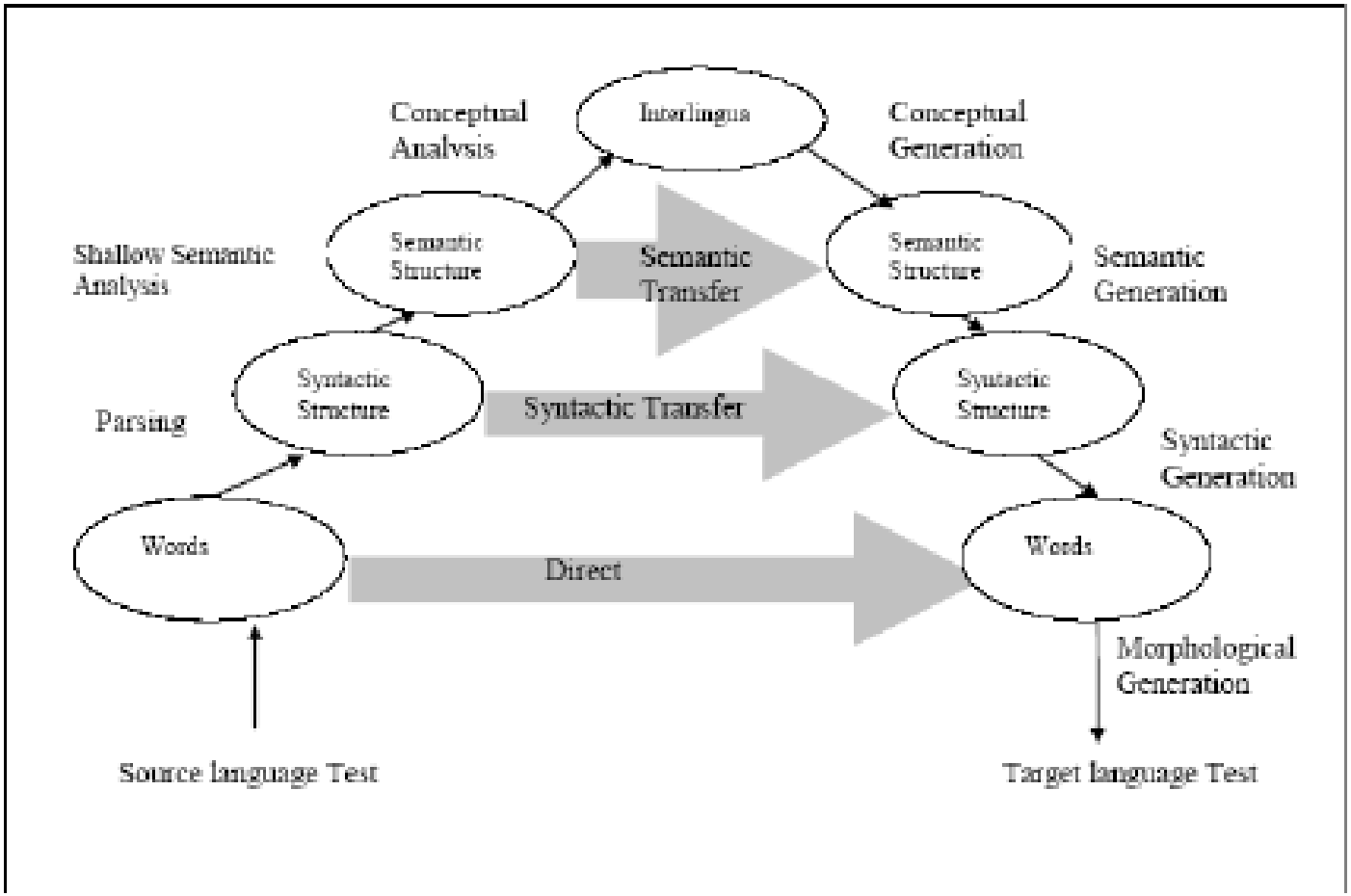
Where is the Louvre Museum located?	In Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	The yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What is the telephone number for Stanford University?	650-723-2300

Traducción Automática

- Grandes promesas con grandes fracasos
- Es mucho más difícil de lo originalmente esperado por varias razones:
  - Morfológicas: Número diferente de morfemas por palabra (uno solo - cantonés, muchos - esquimal, aglutinantes - turco, fusionados - ruso)
  - Sintácticas: Orden de las palabras - Sujeto, Verbo, Objeto (SVO - Francés, Inglés, ... o VSO - Hindi, Japonés, ..., VSO - Árabe, ...)
  - Cómo se ligan las palabras (e.g., the man's house vs. az ember ház-a ó "the man house his" - Húngaro)
  - Con/sin pronombre (e.g., Mostró vs. He showed)
  - Orden de las palabras (e.g., blue house vs. casa azul)
  - Distinta ambigüedad en las palabras o palabras inexistentes!!

Traducción Automática

- Los enfoques tradicionales buscaban hacer traducciones a diferentes niveles de abstracción



Traducción Automática

- Hacer traducción automática perfecta (fiel al significado y natural en la otra lengua) es imposible
- Podemos usar una estrategia Bayesiana

$$\hat{Esp} = argmax P(Esp|Fra)$$

$$\hat{Esp} \approx argmax \overbrace{P(Fra|Esp)}^{\text{modelo traducción}} \overbrace{P(Esp)}^{\text{modelo lenguaje}}$$

- Por lo que se necesitan 3 elementos, el modelo de lenguaje, el modelo de traducción y algo que produzca la traducción más probable

The Phrase-Based Translation Model

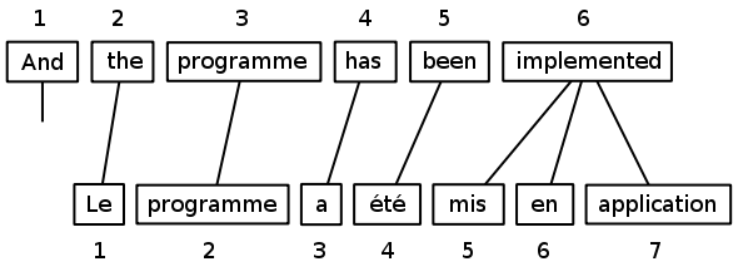
- El modelo de traducción tiene que asignar una probabilidad a cada traducción
- En lugar de pensar en palabras, lo normal es trabajar sobre frases completas
- Se usan: (i) probabilidad de traducción y (ii) probabilidad de distorsión (basada en distancia)

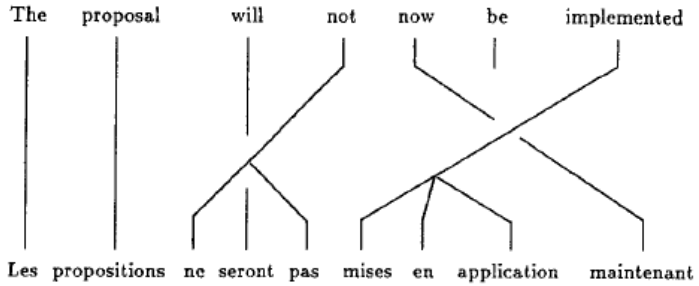
$$P(F|E) = \prod_{i-1}^{I} \phi(\vec{e}_i, \vec{f}_i) d(a_i - b_{i-1})$$

- Las palabras, por ejemplo en español, se agrupan en frases $(\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_I)$, se traduce cada una de ellas al, digamos francés $(\vec{f}_i; i \in [1..I])$
- a_i es el inicio de la palabra en francés generada por la i-ésima frase en español $(\vec{e}_i)$ y b_i es el final de la palabra en francés generada por la $i-1$-ésima frase en español

Traducción Automática

- Para entrenar el modelo se requiere de un grupo grande de traducciones (de francés a español) que tengan además información de la correspondencia entre las frases
- El alineamiento de palabras puede ser uno a uno, contener palabras espúrias (sin correspondencia), ser de uno a muchos o de muchos a muchos





Alineamiento Estadístico

El modelo básico (IBM Model 1) sigue estos tres pasos:

1. Seleccionar una longitud de la frase a traducir
2. Seleccionar un alineamiento entre las palabras de los dos lenguajes
3. Para cada posición, selecciona una traducción

Alineamiento Estadístico

1. Suponiendo que conocemos la longitud de la oración (J) y la alineación (A) de las palabras, la probabilidad de la oración en francés (F) dada la oración en español (E) es:

$$P(F|E,A) = \prod_{j=1}^{J} t(f_j|e_{a_j})$$

donde $t(f_x|e_y)$ es la probabilidad de traducir e_y en f_x

2. Si todos los alineamientos de las palabras son igualmente probables (suposición fuerte)

$$P(A|E) = \frac{\epsilon}{(I+1)^J}$$

para I palabras de la fuente y una constante dada (ϵ)

Alineamiento Estadístico

1. Combinando estas dos probabilidades:

$$P(F, A|E) = P(F|E, A) \times P(A|E)$$

$$= \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(f_j|e_{a_j})$$

2. Dada la suposición de independiencia entre palabras vecinas:

$$\hat{A} = argmax_A P(F, A|E)$$

$$= argmax_A \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(f_j|e_{a_j})$$

$$= argmax_{a_j} t(f_j|e_{a_j}), 1 < j < J$$

Alineamiento Estadístico

1. Se han hecho extensiones para mejorar el alineamiento (usando HMM)
2. Mejorar las estimaciones de probabilidad
3. Permitir que una palabra se refiera a más de una en el otro lenguaje
4. Las últimas tendencias es usar *Deep Learning* con un gran corpus de traducciones (e.g., DeepL)

Procesamiento de Lenguaje Natural

- Dada la cantidad de texto que se genera actualmente, NLP ha creado mucho interés
- Analizar información en redes sociales, autoría, sistemas conversacionales, traductores automáticos, ...
- Los sistemas basados en ML con muchos datos han dominado recientemente el área