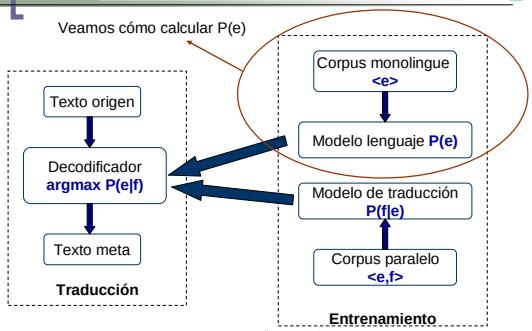
Modelos de lenguaje

- Basados en n-grams
- Sintácticos
- Factorizados
- aleatorios
- Tamaño de los modelos generados

TA estadística



Modelos de lenguaje

- Objetivo: asignar probabilidades a cada posible frase de un idioma
  - Buenas combinaciones de palabras deberían obtener probs altas y frases sin sentido probs bajas
- Intuitivamente se encarga de la fluidez del idioma
  - Fluidez = que el texto generado sea fácil de leer, gramaticalmente correcto y que parezca natural del LM
- Algunas aplicaciones que usan modelos de lenguajes: reconocimiento del habla, reconocimiento de la escritura (handwriting), correcciones ortográficas, OCR, TA, …

Modelos de lenguaje

- Para crea un ML hay que contestar estas preguntas
  1. Cómo sabemos qué frases son posibles en un idioma?
  2. Cómo calculamos P(e)?
- Podría tomar un corpus monolingüe gigante, la web
  - Para 1. podemos tomar cada oración que aparece en la web como una frase posible del LM
  - Para 2. calcular la frecuencia relativa de todas las frases de un idioma (#veces_que_aparece/#total_frases_idioma)
- Problemas de este modelo simplista
  - Muchas frases no aparecen en la web → obtienen 0 sean correctas o no en LM

Modelos basados en n-grams

- Una mejor opcion es partir cada oración observada en n-gramas
  - Hipótesis: si una oración contiene muchos n-gramas probables → más chances de sea una buena frase en LM
- Los ML basados en n-gramas asumen que la k-ésima palabra depende de las k-1 anteriores (historia)
  - $P(w_1^k) = P(w_1)*P(w_2|w_1)*\ldots*P(w_k|w_{k-1})$
  - En $P(w_k|w_{k-1})$: w_k es la predicción y w_{k-1} la historia
- Un estimador usado comúnmente es $P_\theta(w|h) = count(h,w)/count(h)$
  - count = #ocurrencias en corpus entrenamiento

Modelos basados en n-grams

- Estos modelos describen el lenguaje como cadenas de Markov de orden n-1

$Pr(w|h) = Pr(w_n|w_1, w_2, \ldots, w_{n-1}) \approx Pr(w_n|w_{n-N+1}, \ldots, w_{n-1})$

- Cadena de Markov = serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior.
- Se dice que tienen memoria: “recuerdan" el último evento y esto condiciona las posibilidades de los eventos futuros.
  - Esto las distingue de eventos independientes, como tirar una moneda al aire o un dado.

Modelos basados en n-grams

- Dado un vocabulario de tamaño *V* los parámetros libres que deben estimarse son V^n-1
  - Por ej con V=10 un modelo por palabras o 1-gram tiene 9 p.l, 2-gram tiene 99 pl, 3-grams tiene 999, etc
- La elección de n influye en la performance del modelo
  - Con n grande el modelo es más preciso pero los estimadores (EMV) son menos confiables
- Otro problema de elegir n muy grande es la baja densidad (data sparseness) de n-grams de orden alto
  - Es decir, habrá muchos n-grams nunca vistos
- Un valor aceptable es n = 3
  - Intuitivamente, genero palabras y sólo recuerdo las dos anteriores que se generaro

Ejemplo

- Supongamos que tenemos la frase x="el perro ladra mucho"
  - P(x) = P(el | inicio_frase inicio_frase) *
    P(perro | inicio_frase el) *
    P(ladra | el perro) *
    P(mucho | perro ladra) *
    P(fin_frase | ladra mucho) *
    P(fin_frase | mucho fin_frase)
- Si alguno de estos n-grams no aparece en el corpus de entrenamiento la oración recibe prob 0
  - → Smoothing

Smoothing

- Smoothing = suavizar → la distribución de probabilidades
  - Re-distribuir masa de n-grams del corpus entre n-grams que no ocurren en corpus
    - "Discount coefficients" se restan de la frecuencia relativa
- Back-off: si $w_1,\dots w_n$ no aparece en el corpus $P(w_n| w_1,\dots w_{n-1})$ se estima con $P(w_n| w_2,\dots w_{n-1})$
- Good-Turing estimation: agrupa las palabras por su frecuencia en el corpus (#ocurrencias) y calcula $P(X) = r^*/N$ con $r^* = (r+1)*E(N_{r+1})/E(N_r)$
  - r: #veces que se vio X, N: tamaño corpus, N_r: #palabras vistas r veces, E(a): esperanza de a, r*: discount coefficient
  - Estima la probabilidad de que la próxima palabra que veamos sea X luego de haber visto un cierto corpus

Modelos basados en n-grams

- Simple linear interpolation: la idea es que modelos de n-grams de menor orden son menos dispersos (sparse) entonces los usa para calcular los de orden n
  - $P(w_n| w_1,\dots w_{n-1}) = \lambda_1 P(w_n) + \lambda_2 P(w_n|w_{n-1}) +\dots+\lambda_n P(w_n| w_1,\dots w_{n-1})$
    - $0<= \lambda_i <=1$, $\Sigma \lambda_i =1$ → los λ_i se calculan con EM
- Katz Backoff: también usa n-grams de menor orden pero sólo uno a la vez
  - Descuenta menos a modelos más frecuentes
  - Smoothing se activa cuando aparece un n-gram desconocido y ahí se elije el modelo con
- General lineal interpolation: combina KB y SLI donde λ_i nn son fijos sino una función de la historia

Modelos basados en n-grams

- Existen otras estrategias de smoothing y combinaciones, por ej caching, skipping, clustering, Kneser-Ney,…
- Investigación en estas técnicas mejora la calidad de los MLs generados pero no resuelven algunos problemas básicos
  - Por ej que el tamaño de N (-gram) está muy limitado por el corpus usado
- Sin embargo estos son los más usados
  - De hecho, las herramientas libres más usadas son CMU y SRI, ambas implementando 2/3-grams (SRI tiene otros mas)

Modelos sintácticos

- Usan gramáticas PCFG para estudiar cómo se relacionan las palabras de un corpus
  - Son Probabilistic context-free grammars entrenadas sobre un corpus de árboles para aprender las probs de aplicar las reglas en oraciones nuevas (plain text)
- Las PCFG se usan para modelar lenguaje y traducir a la vez
  - Los trabajos más conocidos (Charniak 2003) usan PCFG para traducción "tree-to-string"
  - Toman un árbol de parseo del LO, le aplican algunas operaciones (reordenamiento, borrado/inserción de nodos) y traducen las hojas del árbol al LM
- La hipótesis es que la traducción sea mas gramatical → mejor calidad
  - Al depender de parsers y treebanks no es posible aplicarlo a cualquier par de idiomas

Modelos factorizados

- Son una solución intermedia: extensión de los basados en n-grams y menos demandantes (en recursos) que los modelos sintácticos
- Tienen la opción de incorporar conocimiento lingüístico como POS tags, clases semánticas, etc
- En estos modelos una palabra w es una colección de características o factores (incuyendo la palabra misma o surface form)
- Las palabras se representan como vectores de características
  - Ej la = ("la", artículo), gato = ("gato", sustantivo), …

Modelos factorizados

- Palabra w con K características representada por

$$w \equiv \{f^1, f^2, \ldots f^K\} = f^{1:K}$$

- Dada una historia de n-1 palabras, la prob de que la próxima sea w_t es

$$P(w_1, w_2, \ldots, w_T) = P(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K}) = P(f_{1:T}^{1:K})$$

- Estos modelos simulan la aplicación de varios modelos n-gram cada uno con sus parámetros y estrategias de smooth

Modelos aleatorios

- Siguen siendo basados en n-grams pero en vez de guardar explícitamente cada n-gram distinto guardan un muestreo aleatorio
  - Usan filtros de Bloom para representar el muestreo con arrays de bits
- Intuitivamente, modela el hecho de alguna información se pierde en el proceso de tradución
  - Es decir, que podemos equivocarnos en un porcentaje ε
- La gran ventaja de estos modelos es la reducción del espacio de memoria necesario para guardar el ML

Tamaño del ML

- Los ML pueden ser enormes dependiendo de la cantidad de textos usado
  - Ej [Osborne 2007] muestra la cantidad de n-grams distintos en dos corpora grandes

Corpus	Europarl	Gigaword
1-gms	61K	281K
2-gms	1.3M	5.4M
3-gms	4.7M	275M
4-gms	9.0M	599M
5-gms	10.3M	842M
6-gms	10.7M	957M

Table 2: Number of distinct n-grams

Tamaño del ML

- Durante el proceso de traducción el ML se utiliza muchas veces por oración pero no es posible alojarlos en memoria en una sola máquina
  - Según algunas estimaciones un ML 5-gram usando 2 billones de palabras no podría estimarse en una maquina con 50 GB RAM
- Podríamos filtrarlos por ej descartando n-grams con probs menor a un cierto imite (cut-off)
  - Estas estrategias ayudan pero también afectan la calidad
- Por lo tanto, queda usar el disco (swap) o distribuir entre varias maquinas

Tamaño del ML

- El uso de grandes MLs es una rea de investigación bastante activa motivada por la necesidad de mejorar la calidad final de las aplicaciones de PLN
- Ejemplo: Google [Osborne 2010]
  - Entrenado con 2 trillones de tokens
  - Usando 1,500 maquinas generaron su ML en 1 día
  - ML alojado en un cluster → Probabilidades del ML se piden a través de una red
  - Y ya vemos que esto puede ayudar bastante a la calidad de las traducciones !

Evaluación de MLs

- Para comparar distintos MLs es necesario establecer algunas métricas que indiquen la calidad de los mismos
- La más conocida es *perplexity (PP)* aunque también puede usarse la *entropía* del modelo
- La *PP* de un ML respecto a un corpus S de |S| palabras es $Pr(S)^{-1/|S|}$
  - El ML con menor PP será el que asigne mayor probs a S
- Por lo tanto un ML es mejor que otro si su PP es menor

Evaluación de MLs

- La PP depende de dos cosas: el ML y el corpus
  - Como función del ML mide qué tan bien se ajusta al lenguaje modelado
  - Como función del corpus mide la complejidad del idioma en cuestión
- En la práctica
  - Es importante que se use un corpus representativo de la aplicación pensada para el ML
  - Una reducción de la PP del 10% se considera significante

Bibliografía

- Randomized LMs
  http://www.inf.ed.ac.uk/teaching/courses/mt/papers/randlm.pdf
- Google's LM
  http://www.inf.ed.ac.uk/teaching/courses/mt/lectures/lm.pdf
- Syntax-based LMs
  http://list.cs.brown.edu/research/pubs/pdfs/2003/Charniak-2003-SBL.pdf
- Factored LMs
  http://www.iccs.informatics.ed.ac.uk/~miles/phd-projects/axelrod.pdf
- N-gram LMs
  http://acl.ldc.upenn.edu/J/J92/J92-4003.pdf
- CMU toolkit
  http://www.speech.cs.cmu.edu/SLM/toolkit.html
- SRI Language Model
  http://www-speech.sri.com/projects/srilm/