Language Model a N-grammi

CORSO DI NATURAL LANGUAGE PROCESSING (ELABORAZIONE DEL LINGUAGGIO NATURALE)

a.a. 2025/2026

Prof. Roberto Pirrone

Modelli del Linguaggio probabilistici

- Obiettivo: assegnamo una probabilità a una frase
  - Machine Translation:
    - P(**high** winds tonite) > P(**large** winds tonite)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Summarization, question-answering, ….

Perché?

Modelli del Linguaggio probabilistici

- Calcoliamo la probabilità di una frase come sequenza di parole:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- Ovvero: la probabilità della parola successiva:

$$P(w_5|w_1,w_2,w_3,w_4)$$

- Un moello che calcoli una di queste probabilità:

$$P(W) \text{ or } P(w_n|w_1,w_2 \ldots w_{n-1})$$ è detto **language model**.

Meglio: **la grammatica** ma **language model** o **LM** è standard

Come calcolare P(W)

- Calcoliamo la probabilità congiunta:
  - P(its, water, is, so, transparent, that)
- Appoggiamoci alla Chain Rule della Probabilità

La Chain Rule applicata alla probabilità congiunta delle parole di una frase

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

Come stimare queste probabilità

- Semplicemente contando?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Troppe frasi possibili!
- Non abbiamo abbastanza dati

Assunzione di Markov


Andrei Markov

- Ricordiamo che:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- O magari

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

Il più semplice LM: il modello a Unigrammi

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Un esempio di frasi generate automaticamente dal modello

```
fifth, an, of, futures, the, an, incorporated, a, a,
the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Modello a bigrammi

Condizioniamo sulla parola precedente:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

Modelli a N-grammi

- Possiamo estendere ai trigrammi, 4-grammi, 5-grammi …
- In generale, questo è un LM insufficiente
- Il linguaggio ha **dipendenze a lunga distanza**:

> “The computer which I had just put into the machine room on the fifth floor crashed.”

Stima delle probailità dei bigrammi

- Usiamo la MLE e, trattandosi di variabili stocastiche discrete, non possianmo far altro che contare le loro occorrenze nel data set

$$P(w_{i-1}, w_i) = \frac{c(w_{i-1},w_i)}{\sum_w c(w_{i-1},w\,)} = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$$

Un esempio

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

Token speciali di inizio e fine frase che ci servono per le parole estreme

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

Il corpus Berkeley Restaurant Project (esempi)

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Conteggio dei bigrammi

- 9222 frasi nel data set

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Probabilità dei bigrammi

- Normalizziamo rispetto ai conteggi degli unigrammi:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- E otteniamo:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Stima della probabilità di una frase

P(<s> I want English food </s>) =

P(I|<s>)	= 0.25
× P(want|I)	x 0.33
× P(English|want)	x 0.0011
× P(food|English)	x 0.5
× P(</s>|food)	x 0.68
=	= .000031

Che conoscenza linguistica abbiamo?

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- P(I | <s>) = .25
- P(</s> | food) = .68

Calcoliamo

P(<s> I eat Chinese food </s>)

In pratica …

- Facciamo tutto nello spazio logaritmico.
  - Evitiamo l'underflow.
  - (Inoltre, l'addizione è più veloce della moltiplicazione).

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Google N-Gram, August 2006

AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google Book N-grams

- https://books.google.com/ngrams/

Valutazione: quanto è valido il nostro modello?

- Il nostro modello linguistico preferisce le frasi corrette a quelle errate?
  - Assegna una probabilità maggiore alle frasi “reali” o “frequentemente osservate” rispetto alle frasi “grammaticalmente scorrette” o “raramente osservate”?

- Addestriamo i parametri del nostro modello su un set di addestramento.
- Testiamo le prestazioni del modello su un test set di
- Una metrica di valutazione ci dice quanto è efficace il nostro modello sul set di test.

Valutazione “estrinseca” dei modelli N-gram

- Migliore valutazione per il confronto tra i modelli A e B
- Inserisci ogni modello in un task
  - correttore ortografico, riconoscitore vocale, sistema di traduzione automatica
- Esegui il task ottenendo una misura di accuratezza
- Confronta le accuratezze di A e B

Problemi della valutazione estrinseca

- La valutazione estrinseca è time consuming
- Allora
  - Meglio usare una valutazione **intrinseca**: **perplexity**

  - Una cattiva approssimazione dell'accuratezza
  - I dati di test devono somigliare a quelli di training
  - Ma è un primo passo

Perplexity

Il miglior modello linguistico è quello che meglio prevede un set di test non visto.

Fornisce la P(sentence) più alta

La Perplexity l'inverso della probabilità del test set normalizzata rispetto al numero delle parole:

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

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

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

Unigrami: $$\mathrm{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i)}}$$

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

Minimizzare la perplexity è lo stesso che massimizzare la probabilità

Calcoliamo la perplexity

- La Perplexity è il *branching factor medio pesato* di un linguaggio
  - Branching factor: il numero di possibili parole successive a una data parola in un linguaggio

- Supponiamo di voler valutare il test set $T = \text{“red red red red blue”}$
  - Il nostro “linguaggio” $L = \{\text{red}, \text{blue}, \text{green}\}$
  - In *L* una parola può essere seguita da una qualsiasi altra parola
  - Branching factor = 3

Calcoliamo la perplexity

Supponiamo di voler valutare il test set

$$T = \text{“red red red red blue”}$$

Qual’è la perplexity della frase con un modello che assegna P=1/3 ad ogni parola?

$$\begin{aligned}
\text{perplexity}_A(T) &= P_A(\texttt{red red red red blue})^{-\frac{1}{5}} \\
&= \left(\left(\frac{1}{3}\right)^5\right)^{-\frac{1}{5}} \\
&= \left(\frac{1}{3}\right)^{-1} = 3
\end{aligned}$$

Calcoliamo la perplexity

Training set:

100 parole in *L*

“red” occorre 80 volte

“green” e “blue” occorrono 10 volte ciascuna

$$P(\texttt{red}) = 0.8 \quad P(\texttt{green}) = 0.1 \quad P(\texttt{blue}) = 0.1$$

$$\begin{aligned} \text{perplexity}_B(T) &= P_B(\texttt{red red red red blue})^{-1/5} \\ &= 0.04096^{-\frac{1}{5}} \\ &= 0.527^{-1} = 1.89 \end{aligned}$$

Bassa perplexity → miglior modello

- Wall Street Journal corpus: Training 38 milioni di parole, Test 1.5 milioni di parole

Ordine N-grammi	Unigrammi	Bigrammi	Trigrammi
Perplexity	962	170	109

Un modello a bigrammi a lavoro

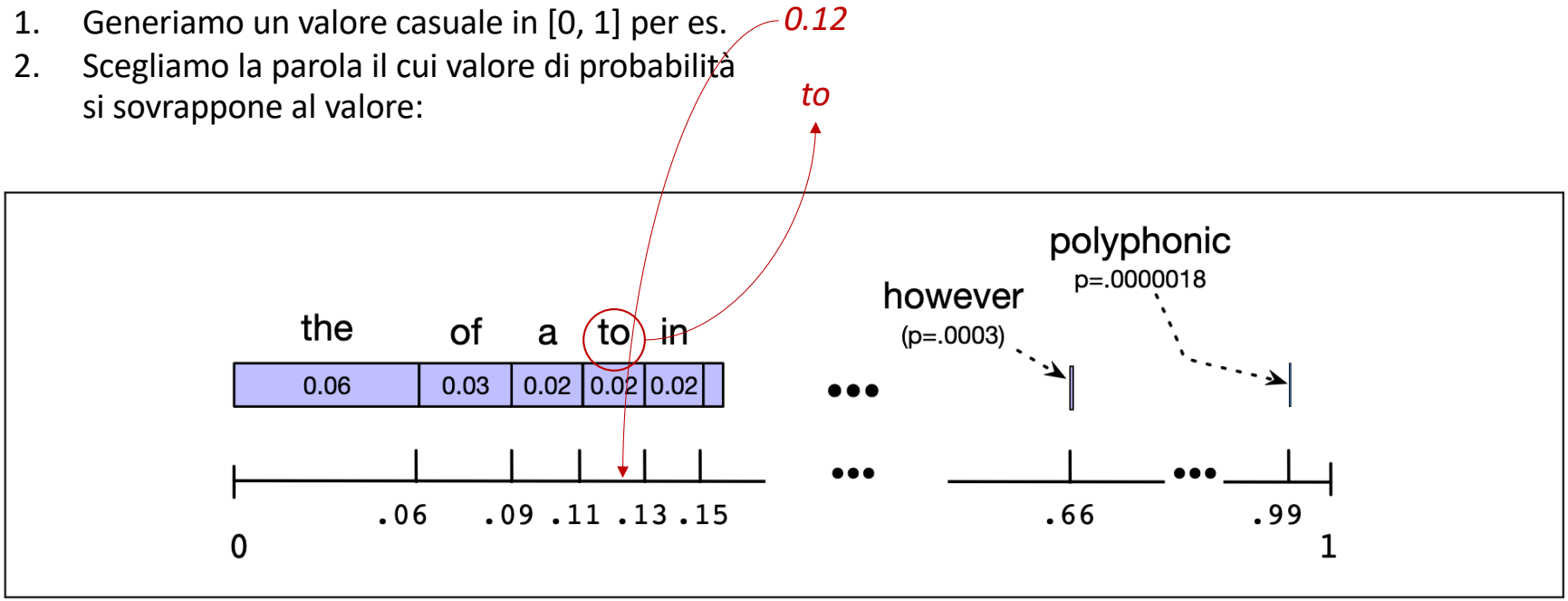
- Scegliamo un bigramma (<s>, w) a caso secondo la sua probabilità
- Scegliamo un bigramma (w, x) a caso secondo la sua probabilità
- E così via finché scegliamo </s>
- E componiamo la stringa

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Che vuol dire: «a caso secondo la sua probabilità» ?

La visualizzazione del campionamento di Shannon

1. Generiamo un valore casuale in [0, 1] per es.
2. Scegliamo la parola il cui valore di probabilità si sovrappone al valore:



Estendibile agli n-grammi

Approssimiamo il corpus di Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

Corpus di Shakespeare

- N=884,647 tokens, V=29,066
- Shakespeare genera 300,000 tipi di bigrammi da V^2= 844 milioni di possibili bigrammi.
  - Quindi il 99.96% dei possibili bigrammi non si sono mai visti (hanno conteggio pari a zero)
- I 4-grammi sono peggio: Ciò che viene campionato sembra Shakespeare perché ***è*** Shakespeare

Il Wall Street Journal non èShakespeare (N = 40 milioni di parole)

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Riassumendo

- Gli N-grammi funzionano meglio all'aumentare di N
  - Ci sono più possibilità di continuare una frase partendo dagli stessi token.
  - Il modello genera frasi non sovrapposte.

- Ma ci sono dei problemi relativi a:
  - Sparsità.
  - Overfitting.

I pericoli dell'overfitting

- Gli N-grammi funzionano bene se il corpus di test è simile al corpus di addestramento.
  - Nella realtà, spesso non è così.
  - Dobbiamo addestrare modelli robusti che generalizzino!
  - Un tipo di generalizzazione: gli zeri!
    - Elementi che non compaiono mai nel set di addestramento, ma compaiono nel set di test.

Zeri

- Training set:
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request

  P(“offer” | denied the) = 0

- Test set
  - … denied the offer
  - … denied the loan

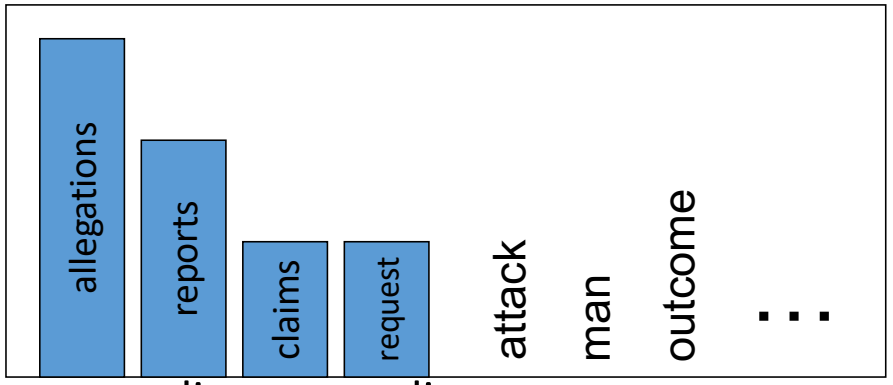
Bigrammi con zero probabilità

- Assegnano probabilità 0 alle frasi del test set
- Non possiamo calcolare la perplexity (divisione per 0)

L’intuizione dello smoothing

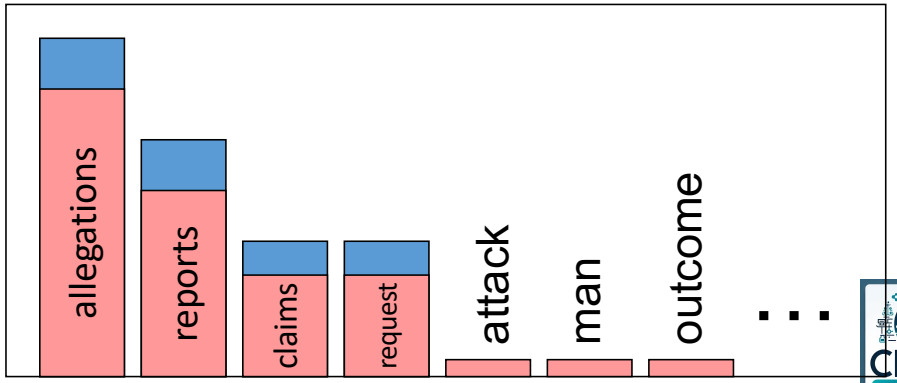
- Abbiamo una statistica sparsa:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Rubiamo un po’ di probabilità di massa per generalizzare meglio

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Stima Add-1 (modifichiamo un po' la MLE)

- Chiamato anche smoothing di Laplace
- Facciamo finta di aver visto ogni parola una volta in più rispetto a quanto abbiamo fatto realmente
- Aggiungiamo 1 a tutti I conteggi!

- Stima MLE:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Stima Add-1:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

$P_{Add-1} \rightarrow 1/V$ when $c(w_{i-1}) \rightarrow 0$

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

Berkeley Restaurant Corpus: conteggio dei bigrammi con smoothing di Laplace

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigrammi con smoothing di Laplace

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Ricostruzione dei conteggi

$$c^*(w_{n-1}w_n) = P^*(w_{n-1}w_n)N$$
$$= P^*(w_n|w_{n-1})P(w_{n-1})N$$
$$= P^*(w_n|w_{n-1})C(w_{n-1})$$

$$= \frac{[C(w_{n-1}w_n) + 1]C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Confrontiamo con I conteggi grezzi

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

La stima Add-1 è uno strumento poco preciso

- Non è usato per gli N-grammi:
  - Useremo metodi migliori
- Ma l'add-1 viene usato in altri modelli NLP
  - Text classification
  - Dove la sparsità è moderata.
- Potremmo usare una strategia **add-*k***
  - *k* <1 (0.5, 0.01, …)

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

Backoff e Interpolazione

- A volte è utile usare meno contesto
  - Condizioniamo il contesto a un contesto di cui non abbiamo imparato molto
- **Backoff:**
  - Usiamo i trigrammi se ne abbiamo una buona evidenza,
  - altrimenti bigrammi, altrimenti unigrammi
- **Interpolation:**
  - mixiamo unigrammi, bigrammi, trigrammi

- L'interpolazione lavora meglio

Interpolazione lineare

- Schema semplice

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

$$\sum_i \lambda_i = 1$$

- I lambda dipendono dal contesto:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) \\ &+\lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

Come calcoliamo i lambda?

- Usiamo un validation set

- Scegliamo i λ per massimizzare la probabilità del validation set:
  - Calcoliamo la probabilità degli N-grammi sul training set
  - Con quelle fissate, massmizziamo le probabilità sul validation set in funzione dei λ:

$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

Expectation Maximization

Absolute discounting: sottraiamo un po' da tutti l conteggi

- Supponiamo di voler sottrarre un po' da un conteggio di 4 per calcolare la probabilità di massa per gli zeri.
- Quanto sottraiamo?
- Church and Gale (1991) ebbero un'idea
- Usarono il dataset AP Newswire da 22Mwords
  - Training e validation set
  - Per ogni bigramma nel training set
  - Cerchiamo il suo conteggio nel validation set!!

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Risultato c = (c - .75)*

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

Interpolazione con Absolute Discounting

- Sottraiamo esattamente 0.75!! (o in certo valore *d*)

Bigramma discounted

Peso di interpolazione

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigramma

(magari valutiamo *d* diversamente per i conteggi 1 e 2)

- Siamo certi che è ok usare direttamente l'unigramma P(w)?

Smoothing di Kneser-Ney Smoothing I

- Stima migliore delle probabilità degli unigrammi di ordine inferiore!
  - Shannon game: *I can't see without my reading*_____?
  - "Kong" è in realtà più comune di "glasses"
  - … ma "Kong" segue sempre "Hong"

- Allora invece di P(w): "Quanto è probabile *w*" usiamo
- $P_{continuation}$(w): "Quanto è probabile che *w* appaia come continuazione?"
  - Per ogni parola, contiamo il numero di tipi di bigrammi che completa.
  - Ogni tipo di bigramma è una nuova continuazione la prima volta che viene visto.

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

Smoothing di Kneser-Ney Smoothing II

- Quante volte w appare come una nuova continuazione:

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

- Normalizziamo per il numero totale dei diversi *tipi dei bigrammi*

$$\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|$$

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

Smoothing di Kneser-Ney III

- Alternativamente: Il numero dei *tipi* che abbiamo visto precedere *w*

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- Normalizzato per il numero di parole che *precedono qualunque altra parola*:

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\sum_{w'} \left|\{w'_{i-1} : c(w'_{i-1}, w') > 0\}\right|}$$

Una parola frequente (Kong) che ricorre in un solo contesto (Hong) avrà una bassa probabilità di continuazione.

Smoothing di Kneser-Ney IV

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ è una costante di normalizzazione; la probailità di massa che abbiamo scontato

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{ w : c(w_{i-1}, w) > 0 \} \right|$$

Discount normalizzato

Il numero di parole che possono seguire w_{i-1}
= # dei tipi che abbiamo scontato
= # di volte che abbiamo applicato il normalized discount

Università degli Studi di Palermo
dipartimento di ingegneria unipa
CHILAB

Parole sconosciute: compiti Open Vocabulary vs Closed Vocabulary

- Se conosciamo in anticipo tutte le parole
  - Il vocabolario V è fissato
  - Task Closed Vocabulary

- Ma spesso non è così
  - **Out Of Vocabulary** → parole OOV
  - Task Open Vocabulary

Unknown words: Open versus closed vocabulary tasks

- Usiamo invece I token per el parole sconosciute <UNK>
  - Addestreremo sulle probabilità di <UNK>
    - Creiamo un lessico fisso L di dimensione V
    - Nella fase di normalizzazione del testo, Ogni parola non in L diviene <UNK>
    - E possiamo addestrare il modello sulla sua probabilità
  - In decodifica
    - Usiamo le probabilità di <UNK> per ogni parola di test non nel traininig set

Gestione di corpora di N-grammi enormi

- Per esempio il corpus Google N-gram
- Pruning
  - Usiamo solo gli N-grammi con conteggio superiore ad una soglia.
    - In questo modo rimuoviamo gli n-grammi rari di ordine elevato
  - Pruning basato sull'entropia
- Efficienza
  - Strutture dati efficienti come gli alberi o I filtri di Bloom
  - Conserviamo le parole tramite indici e non stringhe
  - Quantizziamo le probabilità (4-8 bits instead of 8-byte float)

Smoothing di corpora di N-grammi enormi

- "Stupid backoff" (Brants *et al*. 2007)
- Nonn usiamo il discounting, ma le frequenze relative

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if} \quad \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4 S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

Riassumendo

- Add-1 smoothing:
  - OK per la categorizzazione del testo, non per il Language Modeling
- Il metodo più comunemente usato:
  - Interpolazione estesa Kneser-Ney
- Per corpora di N-grammi molto grandi:
  - Stupid backoff