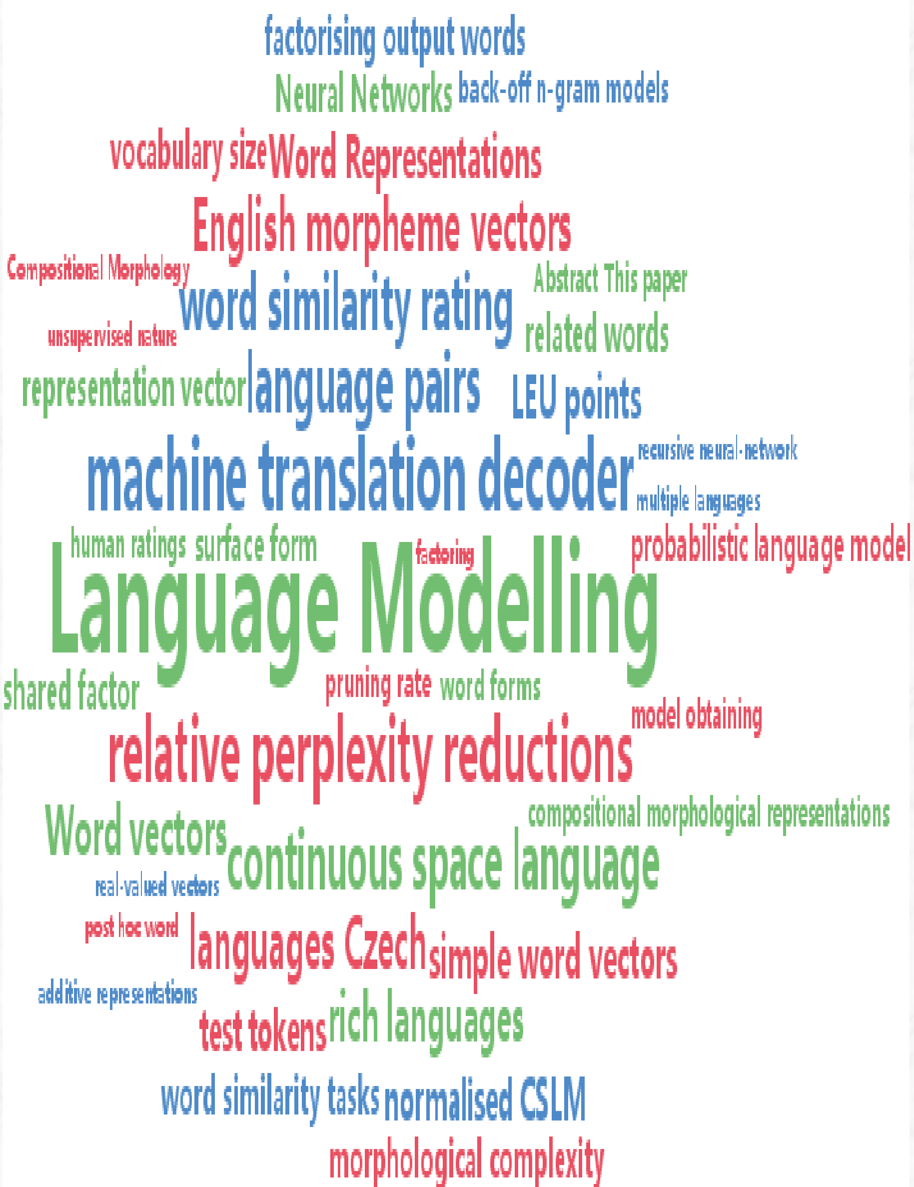
Language Modeling



N-Gram
Chapter 6

Instructor : Dr. Hanaa Bayomi Ali
Mail : h.mobarz @ fci-cu.edu.eg

Language Models

- Formal grammars (e.g. regular, context free) give a hard "binary" model of the legal sentences in a language.
- For NLP, a ***probabilistic*** model of a language that gives a probability that a string is a member of a language is more useful.
- To specify a correct probability distribution, the probability of all sentences in a language must sum to 1.

Probabilistic Language Models

- **The goal: assign a probability to a sentence**

- Speech recognition was the original motivation.
(Related problems are optical character recognition, handwriting recognition.)

  - P(I saw a van) >> P(eyes awe of an)

- Machine Translation:

  - P(**high** winds tonight) > P(**large** winds tonight)

- Spelling Correction

The office is about fifteen **minuets** from my house

  - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)

- + Summarization, question-answering, etc., etc.!!

Language models are everywhere

Probabilistic Language Models

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5…w_n)$

- Related task: probability of an upcoming word:

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

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2…w_{n-1})$ is called a **language model**.
- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the **Chain Rule of Probability**

The Chain Rule: General

- The definition of conditional probabilities

  $P(A \mid B) = P(A, B) / P(B)$

  Rewriting: $P(A, B) = P(A \mid B)\, P(B)$

- More variables:

  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule: joint probability in sentence

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

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)

How to estimate these probabilities

- Could we just count and divide?

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

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

- Simplifying assumption:

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

- Or maybe

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

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

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

Simplest case: Unigram model

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

- An example

I am Sam
Sam I am
I do not like green apples
and bananas

No. Words= 14

P(I)= 3 3/14=0.21
P(sam)=2 2/14=0.14

T
H
I
N
K

Question

Which is assigned higher probability by a unigram language model for English?

- P(I like ice cream)
- P(the the the the)
- P(Go to class daily)
- P(class daily go to)

Answer

Which is assigned higher probability by a unigram language model for English?

- P(I like ice cream)
- P(the the the the)
- P(Go to class daily)
- P(class daily go to)

The word "the" is very frequent in English. A unigram language model does not depend on surrounding words, so "the the the the" gets a high probability even though it isn't regularly used.

Bigram model

- Condition on the previous word:

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

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

- An example

I am Sam
Sam I am
I do not like green apples and bananas

No. Words= 14

P(do|I)= 1/3=0.33
P(am|I)=2/3=0.67

THINK

QUESTION 2

If we estimate a bigram language model from the following corpus, **what is P(not|do)?**

I am Sam

Sam I am

I do not like green eggs and ham

Answer

$P(not|do) = C(do,not)/C(do) = 1/1 = 1$

Train and Test Corpora

- A language model must be trained on **a large corpus of text to estimate good parameter values**.
- Model can be evaluated based on **its ability to predict a high probability** for a disjoint (held-out) test corpus (testing on the training corpus would give an optimistically biased estimate).
- Ideally, the training (and test) corpus should be representative of the actual application data.
- May need to ***adapt*** a general model to a small amount of new (***in-domain***) data by adding highly weighted small corpus to original training data.

N-gram models – Problem (1)

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**

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

- But we can often get away with N-gram models

N-gram models – Problem (2)

- How to handle words in the test corpus that did not occur in the training data, i.e. **out of vocabulary** (OOV) words?

- Out of vocabulary words are words that are not in the training set, but appear in the test set, real data. The main problem is that the model **assigns a probability zero** to out of vocabulary words resulting in a zero likelihood. This is a common problem, specially when you have not trained on a smaller data set.

Smoothing

- Since there are a combinatorial number of possible word sequences, many rare (but not impossible) combinations never occur in training, so MLE incorrectly assigns zero to many parameters (a.k.a. ***sparse data***).
- If a new combination occurs during testing, it is given a probability of zero and the entire sequence gets a probability of zero (i.e. infinite perplexity).
- In practice, parameters are ***smoothed*** (a.k.a. ***regularized***) to reassign some probability mass to unseen events.
  - Adding probability mass to unseen events requires removing it from seen ones (***discounting***) in order to maintain a joint distribution that sums to 1.

Laplace (Add-One) Smoothing

- "Hallucinate" additional training data in which each possible N-gram occurs exactly once and adjust estimates accordingly.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)+1}{C(w_{n-N+1}^{n-1})+V}$$

where ***V*** is the total number of possible (N–1)-grams (i.e. the vocabulary size for a bigram model).

More examples: Berkeley Restaurant Project sentences

- 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

Raw bigram counts (absolute measure)

- Out of 9222 sentences

B\A	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

Raw bigram probabilities (relative measure)

- Normalize by unigrams:

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

- Result:

$P(want|i)= C(i,want)/C(i)=827/2533=0.33$

	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

Bigram estimates of sentence probabilities

$P(\text{I want Chinese food}) =$

$\times\ P(\text{want}|\text{I})$

$\times\ P(\text{Chinese}\ |\text{want})$

$\times\ P(\text{food}|\ \text{Chinese})$

$= 0.33 \times 0.0065 \times 0.52$

$=\mathbf{0.00112}$

What kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065

world

- P(to|want) = .66
- P(eat | to) = .28

grammar

- P(food | to) = 0

grammar (contingent zero)

- P(want | spend) = 0

grammar (structural zero)

Practical Issues

- We do everything in log space
  - –Avoid underflow
  - –(also adding is faster than multiplying)

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