Language modeling

Nils Holzenberger

September 12, 2025

Outline

1. Probabilities in NLP (and speech)
   - Random variables
   - Probabilities in NLP
   - Solving NLP with probabilities
2. Language modeling
   - Σ^*
   - Modeling p
   - Building a Language Model
3. LMs in practice
   - Statistical tools
   - Technical tools
   - Alignment

Outline

Probabilities

- What is a probability?

Probabilities

- What is a probability?
  If I toss a coin, what is the probability it lands on tails?
  If I throw two even dice, what is the probability of a double six?

Probabilities

- What is a probability?
  If I toss a coin, what is the probability it lands on tails?
  If I throw two even dice, what is the probability of a double six?
- Frequency of outcome if I toss the same coin 10,000 times?
- Measurement of my belief that the coin will land on tails?
- I'm playing poker and my opponent gets a royal flush three rounds in a row. Is my opponent cheating?

Random variables

A random variable is a function that maps the outcome of an experiment to a value

Coin-flipping experiment:

$X = \{\text{“the coin lands on heads”} \rightarrow X = 1, \text{“the coin lands on tails”} \rightarrow X = 0\}$

We can reason about the probability of $X = 1$, noted $p(X = 1)$

What is the proportion of outcomes that would result in $X = 1$? What's my belief that $X = 1$ before tossing the coin? How much money would I bet on the result $X = 1$?

Random variables

A random variable is a function that maps the outcome of an experiment to a value

Poker game:

$Y = \{\text{"my opponent cheated"} \rightarrow Y = 1, \text{"my opponent did not cheat"} \rightarrow Y = 0\}$

$Z = \{\text{"my opponent is dealt a royal flush"} \rightarrow Z = 1, ...\}$

Random variables

- Random variables are not random
- Random variables are not variables

Random variables

- Random variables are not random
- Random variables are not variables
- Random variables are functions
- Random variables are deterministic

Random variables

- Random variables are not random
- Random variables are not variables
- Random variables are functions
- Random variables are deterministic
- The randomness comes from the outcome
- A random variable deterministically maps an outcome to a value

Adapted from Ryan Cotterell's Introduction to NLP

Probabilities in NLP

One example, spam classification:

Experiment = I receive an email

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

Properties useful in NLP

- Joint probabilities $p(X = x, Y = y) \stackrel{\text{def}}{=} p(\{X = x\} \cap \{Y = y\})$
  X is a question, Y is an answer
- Conditional probabilities $p(Y = y | X = x) \stackrel{\text{def}}{=} \frac{p(X=x, Y=y)}{p(X=x)}$
  probability that the answer is y given that the question is x

Probabilities in NLP

Express the quantities of interest as random variables.

e.g. spam classification:

Experiment = I receive an email

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

Probabilities in NLP

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

$p(y|x)$ "Given that I received email x, is it spam?"

$p(y)$ "How probable is it that an email I receive should be spam?"

$p(x)$ "How probable is it that I should receive email x?"

$p(x|1)$ "How probable is it that I should receive email x, assuming that it's spam?"

$p(x|0)$ "How probable is it that I should receive email x, assuming that it's not spam?"

Probabilities in NLP

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

How to compute $p(y|x)$?

Computing $p(y|x)$

We have examples of emails that are or are not spam:

X	Y
Alzheimer : Ces petits signes qui DOIVENT vous alerter ! Dès 50 ans, votre cerveau perd chaque jour un peu plus de son potentiel...	1
Vendre sa voiture rapidement et au meilleur prix ! La vente de votre voiture : étape par étape...	1
Hi all, sorry for duplicate sending but there will be this seminar today by Petr Kuznetsov...	0

Computing $p(y|x)$

We have examples of emails that are or are not spam:

X	Y
Alzheimer : Ces petits signes qui DOIVENT vous alerter ! Dès 50 ans, votre cerveau perd chaque jour un peu plus de son potentiel...	1
Vendre sa voiture rapidement et au meilleur prix ! La vente de votre voiture : étape par étape...	1
Hi all, sorry for duplicate sending but there will be this seminar today by Petr Kuznetsov...	0

Typically, approximate $p(y|x)$ with $f_\theta(x, y)$ where θ are parameters, estimated from data.

Computing $p(y|x)$

We have examples of emails that are or are not spam:

X	Y
Alzheimer : Ces petits signes qui DOIVENT vous alerter ! Dès 50 ans, votre cerveau perd chaque jour un peu plus de son potentiel...	1
Vendre sa voiture rapidement et au meilleur prix ! La vente de votre voiture : étape par étape...	1
Hi all, sorry for duplicate sending but there will be this seminar today by Petr Kuznetsov...	0

Typically, approximate $p(y|x)$ with $f_\theta(x, y)$ where θ are parameters, estimated from data.

- What form does f have? *Transformer*, *n-gram*, ...
- What data is used? *Difficulty to collect and annotate*, *quantity*, *quality*, *distribution shifts*, *statistical artefacts*, ...
- How is the estimation done? *Random search*, *gradient descent*, ...

$p(y|x)$ as $p(x)$

Outcome space = strings of *acceptable* characters

Random variable: X = the string that came out

All language tasks can be treated as the above problem

$p(\text{“Where is Télécom Paris located? In Palaiseau”}) \approx 1$

$p(\text{“Where is Télécom Paris located? In Paris”}) \approx 0$

Or at least:

$p(\text{“Where is Télécom Paris located? In Palaiseau”}) >$
$p(\text{“Where is Télécom Paris located? In Paris”})$

$\rightarrow$ model $p(x)$ where x is a string

Adapted from Ryan Cotterell's Introduction to NLP

Probabilities in NLP

- NLP tasks can be done entirely without probabilities
- The meaning of the probability of a sentence or string was controversial for a long time
- Remember that this is just a tool[1]

[1] *All models are wrong, but some are useful* (https://en.wikipedia.org/wiki/All_models_are_wrong)

Outline

Language modeling

- Let Σ be the set for all words in the language

Language modeling

- Let Σ be the set for all words in the language
  - $\Sigma = \{A, B, C, ...\}$
  - $\Sigma = \{\text{acrobate}, \text{arrêt}, \text{arrière}, \text{barre}, ...\}$
  - $\Sigma = \{0, 1\}$
- Kleene closure of any combination of elements of Σ:

$$\Sigma^* = \{v \circ w | v \in \Sigma^*, w \in \Sigma\} \cup \{\epsilon\}$$

($\circ$ string concatenation, ϵ the empty string)

Language modeling

$\Sigma = \{\text{Borislav}, \text{Frederica}, \text{in}, \text{lab}, \text{programs}, \text{the}, \text{with}\}$

$$\Sigma^* = \{$$

ϵ,
Borislav lab programs,
Borislav programs in the lab,
Frederica programs in the lab with Borislav,
Borislav Borislav Borislav,
...

$$\}$$

When Σ is the set of all alphanumeric characters, Σ^* contains the proof of the Poincaré conjecture, the correct answers to the final exam of APM_0EL07_TP, wrong answers to the final exam of APM_0EL07_TP, the results of the next election, all of the Internet...

Adapted from Ryan Cotterell's Introduction to NLP

Language model

A *language model* over the language Σ is a probability distribution p over elements of Σ^*

- Experiment = drawing an element from G, a set of acceptable/valid/desirable sentences

Language model

A *language model* over the language Σ is a probability distribution p over elements of Σ^*

- Experiment = drawing an element from G, a set of acceptable/valid/desirable sentences
- Random variable X = the string drawn from G
  Clearly $G \subset \Sigma^*$ so X takes values in Σ^*

Language model

A *language model* over the language Σ is a probability distribution p over elements of Σ^*

- Experiment = drawing an element from G, a set of acceptable/valid/desirable sentences
- Random variable X = the string drawn from G
  Clearly $G \subset \Sigma^*$ so X takes values in Σ^*
- Consequently, for a string $x \in \Sigma^*$:
  - If $x \in G$, $p(X = x) > 0$
  - If $x \notin G$, $p(X = x) =$

Language model

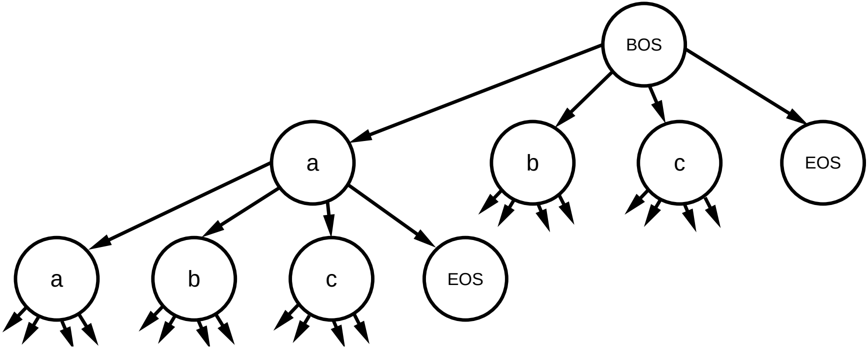
A *language model* over the language Σ is a probability distribution p over elements of Σ^*

- Experiment = drawing an element from G, a set of acceptable/valid/desirable sentences
- Random variable X = the string drawn from G
  Clearly $G \subset \Sigma^*$ so X takes values in Σ^*
- Consequently, for a string $x \in \Sigma^*$:
  - If $x \in G$, $p(X = x) > 0$
  - If $x \notin G$, $p(X = x) = 0$

In practice, Σ is a set of *subword units* : groups of frequent character sequences like *th* or *ing*, which may sometimes be entire words.

Prefix tree

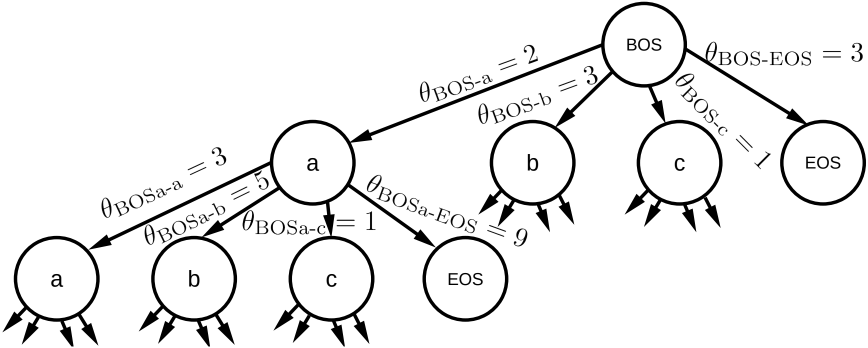
Σ^* can be viewed as a prefix tree:



Adapted from Ryan Cotterell's Introduction to NLP

Prefix tree

A language model is a *weighting* of the prefix tree:

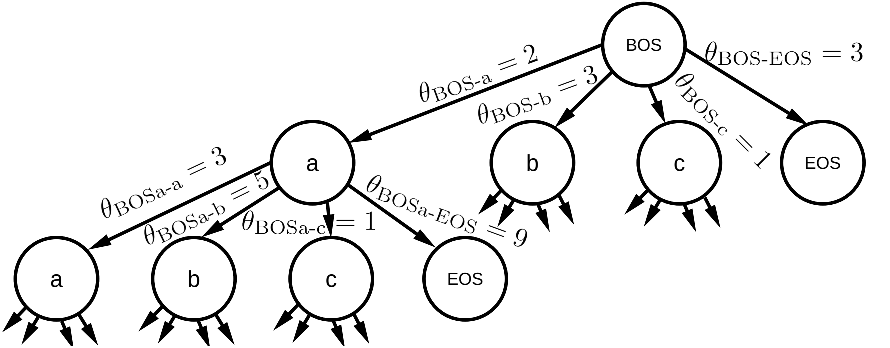


Adapted from Ryan Cotterell's Introduction to NLP

Prefix tree

$\forall x \in \Sigma^*$: $x = x_1 \circ x_2 \circ \ldots \circ x_n$ where $\forall i, x_i \in \Sigma$

$$p(x) = p(x_1 \circ x_2 \circ \ldots \circ x_n) = \frac{1}{Z} \prod_{t=1}^{|x|} \theta_{x_{1:t}} \qquad Z = \sum_{x \in \Sigma^*} \prod_{t=1}^{|x|} \theta_{x_{1:t}}$$



Adapted from Ryan Cotterell's Introduction to NLP

Computing Z

$$Z = \sum_{x \in \Sigma^*} \prod_{t=1}^{|x|} \theta_{x_{1:t}}$$

- Global Normalization
  - No general-purpose method for global normalization
  - Take into account the structure of the probability model (lattice, probabilistic context-free grammar...)
- Local Normalization
  Choose the weights such that $Z = 1$

Adapted from Ryan Cotterell's Introduction to NLP

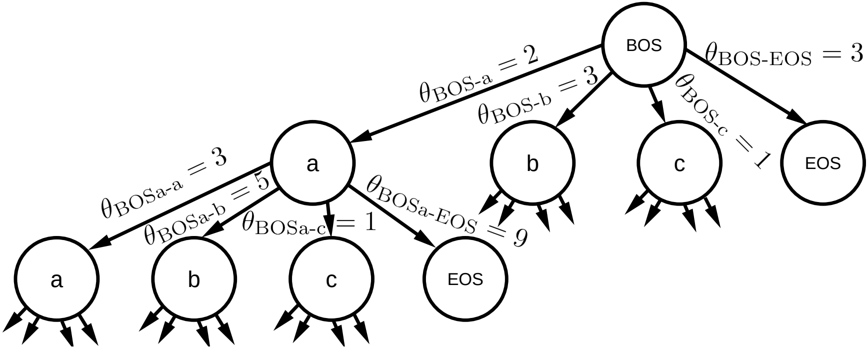
Chain rule

Read it off the tree (it's a graphical model)

$p(\mathrm{BOS\ a\ EOS}) = p(\mathrm{EOS}|\mathrm{BOS\ a})p(\mathrm{a}|\mathrm{BOS})p(\mathrm{BOS})$

$\forall x \in \Sigma^*$:

$p(x) = p(x_1 \circ x_2 \circ \ldots \circ x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1 \circ x_2)\ldots p(x_n|x_1 \circ x_2 \circ \ldots \circ x_{n-1})$



Language modeling

- This is the basic framework of language modeling
- Everything that follows makes some sort of simplifying assumption
  - N-grams
  - Recurrent Neural Networks
- What assumptions is GPT-n making? Is GPT-n a language model? ($n = 1, 2, 3$)

n-gram language modeling

$p(x_t|x_{<t}) \overset{\text{assumption}}{=} p(x_t|x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1})$

e.g. $p(\text{lab}|\text{Frederica programs in the}) = p(\text{lab}|\text{in the})$ $(n =$

n-gram language modeling

$$p(x_t|x_{<t}) \stackrel{\text{assumption}}{=} p(x_t|x_{t-n+1}, x_{t-n+2}, \dots, x_{t-1})$$

e.g. $p(\text{lab}|\text{Frederica programs in the}) = p(\text{lab}|\text{in the})$ (n =3)

- *I only need to look at the* $n-1$ *previous tokens to decide what the next token is*
- 2 assumptions in one:
  - Length of history
  - Absolute position of tokens is irrelevant

n-gram language modeling

$$p(x_t | x_{<t}) \stackrel{\text{assumption}}{=} p(x_t | x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1})$$

- The number of strings to consider is

n-gram language modeling

$$p(x_t|x_{<t}) \overset{\text{assumption}}{=} p(x_t|x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1})$$

- The number of strings to consider is $|\Sigma|^n$
- Assuming 10k tokens in the vocabulary and $n = 10$, that's $10000^{10} - 10000^9 \approx 10^{40}$ free parameters
- Solution: dense representations. Hope that there are inherent constraints in language that make it possible to use fewer free parameters

Density assumption

$$p(x_t | x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1}) \stackrel{\text{def}}{=} \frac{\exp(w_{x_t} \cdot h_t)}{\sum_{z \in \Sigma} \exp(w_z \cdot h_t)}$$

- $w_x \in \mathbb{R}^d$: embedding of token x
- $h_t \in \mathbb{R}^d$: embedding of history

For example:

- $d = 300$
- $w_x \in \mathbb{R}^{300}$
- $h_t \stackrel{\text{def}}{=} \sum_{k=1}^{10} M_k w_{x_{t-k}}$
- $M_k \in \mathbb{R}^{300 \times 300}$

Number of free parameters:

Density assumption

$$p(x_t | x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1}) \stackrel{\text{def}}{=} \frac{\exp(w_{x_t} \cdot h_t)}{\sum_{z \in \Sigma} \exp(w_z \cdot h_t)}$$

- $w_x \in \mathbb{R}^d$: embedding of token x
- $h_t \in \mathbb{R}^d$: embedding of history

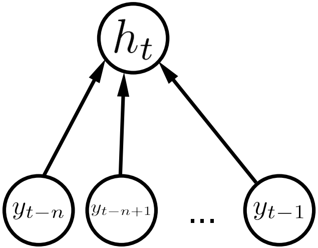
For example:

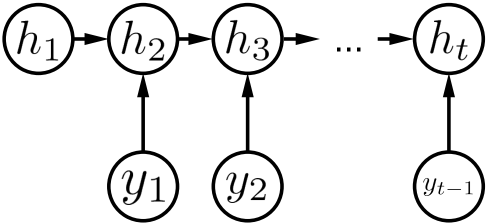
- $d = 300$
- $w_x \in \mathbb{R}^{300}$
- $h_t \stackrel{\text{def}}{=} \sum_{k=1}^{10} M_k w_{x_{t-k}}$
- $M_k \in \mathbb{R}^{300 \times 300}$

Number of free parameters:
$10000 \cdot 300 + 10 \cdot 300^2 = 3 \cdot 10^6 + 9 \cdot 10^5 << 10^{40}$

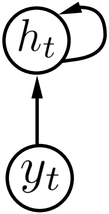
RNN language modeling

- Is the n-gram assumption a major limitation?
- Recurrent Neural Networks can get rid of it
- Keep the density assumption

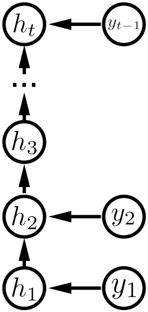




Two views of RNNs



$$h_{t+1} = f(h_t, y_t)$$



$$h_{t+1} = f(f(...f(f(h_1, y_1), y_2)..., y_{t-1}), y_t)$$

An RNN is a very deep neural network with shared parameters between layers

Language modeling

Do RNNs and n-grams work? See lab session

Building a language model

- We don't know the distribution p over G
- We can't describe G other than with a representative sample of G. In practice, $G =$ a selected subset of the Internet
- How to build a language model:
  - collect elements from G and put them in a set D
  - model p with a parametric function f_θ (usually denoted p_θ)
  - modify parameters θ to maximize $\sum_{x \in D} \log(p_\theta(X = x))$

Using a language model for any NLP task

- So far we have a probability distribution over Σ^* that places most weight on acceptable/valid/desirable elements of Σ^*
- For $x, y \in \Sigma^*$, if $p(x) > p(y)$ then x is better than y
- We can use p to answer questions (*inter alia*):
  $\underset{\text{answer}}{\operatorname{argmax}} \; p(\text{“Where is Télécom Paris located?”} \circ \text{answer})$
- Using the chain rule we can complete sequences
- By treating any NLP task as a text completion task, anything can be solved using p

Using a language model for any NLP task

- So far we have a probability distribution over Σ^* that places most weight on acceptable/valid/desirable elements of Σ^*
- For $x, y \in \Sigma^*$, if $p(x) > p(y)$ then x is better than y
- We can use p to answer questions (*inter alia*):
  $\underset{\text{answer}}{\text{argmax}} \quad p(\text{“Where is Télécom Paris located?”} \circ \text{answer})$
- Using the chain rule we can complete sequences
- By treating any NLP task as a text completion task, anything can be solved using p
- Why does this work better than everything else we've had before?

Outline

Machine learning

- Very active AI research area
- Research in NLP is almost inseparable from research in ML
  - ML methods frequently tested on NLP
  - NLP problems led to development of new ML methods

ML cookbook

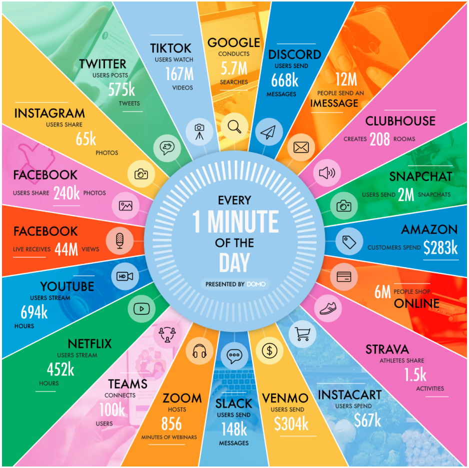
- Language modeling relies on a few simple ideas:
  - A sentence is a random variable
  - A probability density function can be represented by a parametric function (+ density assumption)
  - The real p.d.f. can be approximated by estimating the parameters thanks to the tools of ML
- Many tricks are needed for this to work:
  - The Transformer architecture to model p
  - Language data for G
  - Gradient descent
  - Stability of numerical calculation
  - Hyperparameter search
  - Federated learning, parallel computing
  - Hardware
  - ...

The combustion engine

- The combustion engine relies on basic thermodynamics
  - Carnot cycle
  - Gas physics
  - Thermal reservoirs
- The practice is more complex
  - Materials to build the engine
  - Rotary shaft seals
  - Air-fuel mixture
  - Thermal expansion of the cylinders
  - Lubrication
  - Cooling
  - ...

Internet

- There has never been so much textual data available
- GPT-3 has been trained on around 45TB of text data



DOMO, *Data never sleeps*, `https://www.domo.com/learn/infographic/data-never-sleeps-9`, visité le 2024-03-22-1734

Computing power

- Transformer models only work if trained on large quantities of data
- Transformer models scale well (bigger models work even better, as long as there is enough data to train them)
- This is only practical if computation is fast
- This was made possible by increasingly better GPU cards

One last thing...

- The Internet contains text that is generally syntactically valid
- The Internet contains text of variable semantic quality
  Wikipedia, government websites, Reddit...
- G is curated with heuristics but some stuff always comes through
- LLMs need an *alignment* phase: *Alignment is the process of encoding human values and goals into large language models to make them as helpful, safe, and reliable as possible.*[2]
- In practice
  - let the unaligned LLM interact with humans
  - ask them to flag inappropriate content
  - use the feedback to modify p
  - do this at a massive scale

[2] Jonker and Gomstyn, *What is AI alignment ?*, 2024

Human workforce

- In practice
  - let the unaligned LLM interact with humans
  - ask them to flag inappropriate content
  - use the feedback to modify p
  - do this at a massive scale
- Alignment uses Reinforcement Learning with Human Feedback, a recent theoretical tool
- How much and what kind of human labor is involved is proprietary[3]
- This part is crucial

[3] Clément Le Ludec, Maxime Cornet, Antonio A. Casilli, *The problem with annotation. Human labour and outsourcing between France and Madagascar*, Big Data & Society, 2023; Paola Tubaro, *Learners in the loop: hidden human skills in machine intelligence*, Sociologia del Lavoro, 2022; Antonio A. Casilli, *Waiting for Robots. The Hired Hands of Automation*, 2025

Another view of LLMs

- Newton's laws *explain* Tycho Brahe's observations
- LLMs *explain* the data observed on the Internet
- The motion of planets requires a couple of equations, a handful of constants and calculus
- The language on the internet requires a couple of equations and 175B constants for GPT-3

ANNALS OF TECHNOLOGY

CHATGPT IS A BLURRY JPEG OF THE WEB

OpenAI's chatbot offers paraphrases, whereas Google offers quotes. Which do we prefer?

By **Ted Chiang**

February 9, 2023

Lab session

- The lab contains questions. The questions are repeated in QUESTIONS.txt.
- You will need to write code to answer the questions.
- Fill in the lab session questions via a form on moodle.
- This will be graded.