Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

October 28, 2018

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Probability and Language

Probability and Language

Quick guide to probability theory

Entropy and Information Theory

Probability and Language

Assign a probability to an input sequence

Given a URL: `choosespain.com`. What is this website about?

Input	Scoring function
choose spain	-8.35
chooses pain	-9.88
⋮	⋮

The Goal

Find a good **scoring function** for input sequences.

Scoring Hypotheses in Speech Recognition

From acoustic signal to candidate transcriptions

Hypothesis	Score
the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

Scoring Hypotheses in Machine Translation

From source language to target language candidates

Hypothesis	Score
we must also discuss a vision .	-29.63
we must also discuss on a vision .	-31.58
it is also discuss a vision .	-31.96
we must discuss on greater vision .	-36.09
⋮	⋮

Scoring Hypotheses in Decryption

Character substitutions on ciphertext to plaintext candidates

Hypothesis	Score
Heopaj, zk ukq swjp pk gjks w oaynap?	-93
Urbcnw, mx hxd fjwc cx twxf j bnlanc?	-92
Wtdepy, oz jzf hlye ez vyzh l dpncpe?	-91
Mjtufo, ep zpv xbou up lopx b tfdsfu?	-89
Nkuvgp, fq aqw ycpv vq mpqy c ugetgv?	-87
Gdnozi, yj tjp rvio oj fijr v nzxmzo?	-86
Czjkve, uf pfl nrek kf befn r jvtivk?	-85
Yvfgra, qb lbh jnag gb xabj n frperg?	-84
Zwghsb, rc mci kobh hc ybck o gsqfsh?	-83
Byijud, te oek mqdj je adem q iushuj?	-77
Jgqrcl, bm wms uylr rm ilmu y qcapcr?	-76
Listen, do you want to know a secret?	-25

The Goal

- Write down a **model** over sequences of words or letters.
- **Learn** the parameters of the model from data.
- Use the model to **predict** the probability of new sequences.

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 2: Quick guide to probability theory

Probability: The Basics

- Sample space
- Event space
- Random variable

Probability distributions

- P(X): probability of random variable X having a certain value.
  - $P(X = \text{killer}) = 1.05\text{e-}05$
  - $P(X = \text{app}) = 1.19\text{e-}05$

Joint probability

- P(X,Y): probability that X and Y each have a certain value.
  - Let Y stand for choice of a word
  - Let X stand for the choice of a word that occurs before Y
  - P(X = killer, Y = app) = 1.24e-10

Joint Probability: P(X=value AND Y=value)

- Since X=value AND Y=value, the order does not matter
- P(X = killer, Y = app) ⇔ P(Y = app, X = killer)
- In both cases it is P(X,Y) = P(Y,X) = P('killer app')
- In NLP, we often use numerical indices to express this: P(W_{i-1} = killer, W_i = app)

Joint probability

Joint probability table

W_{i-1}	W_i = app	$\mathrm{P}(W_{i-1},W_i)$
$\langle S \rangle$	app	1.16e-19
an	app	1.76e-08
killer	app	1.24e-10
the	app	2.68e-07
this	app	3.74e-08
your	app	2.39e-08

There will be a similar table for each choice of W_i.

Get $\mathrm{P}(W_i)$ from $\mathrm{P}(W_{i-1},W_i)$

$$P(W_i = \text{app}) = \sum_x P(W_{i-1} = x, W_i = \text{app}) = 1.19e - 05$$

Conditional probability

- $\mathrm{P}(W_i \mid W_{i-1})$: probability that W_i has a certain value after fixing value of W_{i-1}.
- $\mathrm{P}(W_i = \text{app} \mid W_{i-1} = \text{killer})$
- $\mathrm{P}(W_i = \text{app} \mid W_{i-1} = \text{the})$

Conditional probability from Joint probability

$$P(W_i \mid W_{i-1}) = \frac{P(W_{i-1}, W_i)}{P(W_{i-1})}$$

- P(killer) = 1.05e-5
- P(killer, app) = 1.24e-10
- P(app | killer) = 1.18e-5

Basic Terms

- $P(e)$ – *a priori* probability or just *prior*
- $P(f \mid e)$ – *conditional* probability. The chance of f given e
- $P(e, f)$ – *joint* probability. The chance of e and f both happening.
- If e and f are *independent* then we can write
  $P(e, f) = P(e) \times P(f)$
- If e and f are not *independent* then we can write
  $P(e, f) = P(e) \times P(f \mid e)$
  $P(e, f) = P(f) \times$?

Basic Terms

- Addition of integers:

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$$

- Product of integers:

$$\prod_{i=1}^{n} i = 1 \times 2 \times 3 \times \ldots \times n$$

- Factoring:

$$\sum_{i=1}^{n} i \times k = k + 2k + 3k + \ldots + nk = k \sum_{i=1}^{n} i$$

- Product with constant:

$$\prod_{i=1}^{n} i \times k = 1k \times 2k \ldots \times nk = k^n \times \prod_{i=1}^{n} i$$

Probability: Axioms

- P measures total probability of a set of events
- $P(\emptyset) = 0$
- $P(\text{all events}) = 1$
- $P(X) \leq P(Y)$ for any $X \subseteq Y$
- $P(X) + P(Y) = P(X \cup Y)$ provided that $X \cap Y = \emptyset$

Probability Axioms

- All events sum to 1:

$$\sum_{e} P(e) = 1$$

- Marginal probability $P(f)$:

$$P(f) = \sum_{e} P(e, f)$$

- Conditional probability:

$$\sum_{e} P(e \mid f) = \sum_{e} \frac{P(e, f)}{P(f)} = \frac{1}{P(f)} \sum_{e} P(e, f) = 1$$

- Computing $P(f)$ from axioms:

$$P(f) = \sum_{e} P(e) \times P(f \mid e)$$

Probability: The Chain Rule

- $P(a, b, c, d \mid e)$
- We cannot simply remove items from the left of $|$ (verify that it violates the definitions we have given based on sets)
- In this case we can use the chain rule of probability to rescue us
- $P(a, b, c, d \mid e) =$ $P(d \mid e) \cdot P(c \mid d, e) \cdot P(b \mid c, d, e) \cdot P(a \mid b, c, d, e)$
- To see why this is possible, recall that $P(X \mid Y) = \frac{p(X,Y)}{p(Y)}$
  - $\frac{p(a,b,c,d,e)}{p(e)} = \frac{p(d,e)}{p(e)} \cdot \frac{p(c,d,e)}{p(d,e)} \cdot \frac{p(b,c,d,e)}{p(c,d,e)} \cdot \frac{p(a,b,c,d,e)}{p(b,c,d,e)}$
- Use chain rule and simplify:

$$P(a, b, c, d \mid e) = P(d \mid e) \cdot P(c \mid d, e) \cdot P(b \mid c, e) \cdot P(a \mid b, e)$$

Probability: The Chain Rule

- $P(e_1, e_2, \ldots, e_n) = P(e_1) \times P(e_2 \mid e_1) \times P(e_3 \mid e_1, e_2) \ldots$

$$P(e_1, e_2, \ldots, e_n) = \prod_{i=1}^{n} P(e_i \mid e_{i-1}, e_{i-2}, \ldots, e_1)$$

Probability: Random Variables and Events

- What is y in $P(y)$?
- Shorthand for value assigned to a random variable Y, e.g. $Y = y$
- y is an element of some implicit **event space**: $\mathcal{E}$

Probability: Random Variables and Events

- The *marginal probability* $P(y)$ can be computed from $P(x, y)$ as follows:
$$P(y) = \sum_{x \in \mathcal{E}} P(x, y)$$
- Finding the value that maximizes the probability value:
$$\hat{x} = \arg\max_{x \in \mathcal{E}} P(x)$$

Log Probability Arithmetic

- Practical problem with tiny $P(e)$ numbers: underflow
- One solution is to use log probabilities:

$$\begin{aligned}\log(P(e)) &= \log(p_1 \times p_2 \times \ldots \times p_n) \\ &= \log(p_1) + \log(p_2) + \ldots + \log(p_n)\end{aligned}$$

- Note that:

$$x = \exp(\log(x))$$

- Also more efficient: addition instead of multiplication

Log Probability Arithmetic

p	$\log(p)$
0.0	$-\infty$
0.1	-3.32
0.2	-2.32
0.3	-1.74
0.4	-1.32
0.5	-1.00
0.6	-0.74
0.7	-0.51
0.8	-0.32
0.9	-0.15
1.0	0.00

Log Probability Arithmetic

- So: $(0.5 \times 0.5 \times \ldots 0.5) = (0.5)^n$ might get too small but $(-1 - 1 - 1 - 1) = -n$ is manageable
- Another useful fact when writing code ($\log_2$ is *log to the base 2*):

$$\log_2(x) = \frac{\log_{10}(x)}{\log_{10}(2)}$$

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 3: Entropy and Information Theory

Probability and Language

Quick guide to probability theory

Entropy and Information Theory

Information Theory

- Information theory is the use of probability theory to quantify and measure "information".
- Consider the task of efficiently sending a message. Sender Alice wants to send several messages to Receiver Bob. Alice wants to do this as efficiently as possible.
- Let's say that Alice is sending a message where the entire message is just one character a, e.g. $aaaa\ldots$. In this case we can save space by simply sending the length of the message and the single character.

Information Theory

- Now let's say that Alice is sending a completely random signal to Bob. If it is random then we cannot exploit anything in the message to compress it any further.
- The *expected* number of bits it takes to transmit some infinite set of messages is what is called entropy.
- This formulation of entropy by Claude Shannon was adapted from thermodynamics, converting information into a quantity that can be measured.
- Information theory is built around this notion of message compression as a way to evaluate the amount of information.

Expectation

- For a probability distribution p
- **Expectation** with respect to p is a weighted average:

$$
\begin{aligned}
E_p[x] &= \frac{x_1 \cdot p_1 + x_2 \cdot p_2 + \ldots + x_n p_n}{p_1 + p_2 + \ldots + p_n} \\
&= x_1 \cdot p_1 + x_2 \cdot p_2 + \ldots + x_n p_n \\
&= \sum_{x \in \mathcal{E}} x \cdot p(x)
\end{aligned}
$$

- Example: for a six-sided die the expectation is:

$$E_p[x] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \ldots + 6 \cdot \frac{1}{6} = 3.5$$

Entropy

- For a probability distribution p
- **Entropy** of p is:

$$H(p) = -\sum_{x \in \mathcal{E}} p(x) \cdot \log_2 p(x)$$

- Any base can be used for the log, but base 2 means that entropy is measured in bits.
- What is the *expected* number of bits with respect to p:

$$-E_p[\log_2 p(x)] = H(p)$$

- Entropy answers the question: *What is the expected number of bits needed to transmit messages from event space $\mathcal{E}$, where $p(x)$ defines the probability of observing x?*

Entropy

- Alice wants to bet on a horse race. She has to send a message to her bookie Bob to tell him which horse to bet on.
- There are 8 horses. One encoding scheme for the messages is to use a number for each horse. So in bits this would be $001, 010, \ldots$
  (lower bound on message length $=3$ bits in this encoding scheme)
- Can we do better?

Entropy

Horse 1	$\frac{1}{2}$	Horse 5	$\frac{1}{64}$
Horse 2	$\frac{1}{4}$	Horse 6	$\frac{1}{64}$
Horse 3	$\frac{1}{8}$	Horse 7	$\frac{1}{64}$
Horse 4	$\frac{1}{16}$	Horse 8	$\frac{1}{64}$

- If we know how likely we are to bet on each horse, say based on the horse's probability of winning, then we can do better.
- Let p be the probability distribution given in the table above. The entropy of p is $H(p)$

Entropy

$$
\begin{aligned}
H(p) = & \\
= & \; -\sum_{i=1}^{8} p(i) \log_2 \; p(i) \\
= & \; -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{16}\log_2\frac{1}{16} + 4(\frac{1}{64}\log_2\frac{1}{64})\right) \\
= & \; -\left(\frac{1}{2}\times -1 + \frac{1}{4}\times -2 + \frac{1}{8}\times -3 + \frac{1}{16}\times -4 + 4(\frac{1}{64}\times -6)\right) \\
= & \; -\left(-\frac{1}{2} - \frac{1}{2} - \frac{3}{8} - \frac{1}{4} - \frac{3}{8}\right) \\
= & \; 2 \; bits
\end{aligned}
$$

- What is the entropy when the horses are equally likely to win?

$$H(uniform\ distribution) = -8(\frac{1}{8} \times -3) = 3 \; bits$$

Entropy

- e.g., most likely horse gets code 0, next most likely gets 10, and then $110, 1110, \ldots$
  many possible coding schemes, this is a simple code to illustrate number of bits needed for a large number of messages ...
- Assume there are 320 messages (one for each race): code 0 occurs 160 times, code 10 occurs 80 times, code 110 occurs 40 times, code 1110 occurs 20 times, code 11110 occurs 5 times.
- Total number of bits for all messages: 160*len(0) + 80*len(10) + 40*len(110) + 20*len(1110) + 5*len(11110)
- Number of bits: $160*1 + 80*2 + 40*3 + 20*4 + 5*5 = 545$
- Total number of bits per message (per race): $\frac{545}{320} \approx 1.7$ bits (always less than 2 bits)

Perplexity

- The value $2^{H(p)}$ is called the **perplexity** of a distribution p
- Perplexity is the weighted average number of choices a random variable has to make.
- Choosing between 8 equally likely horses (H=3) is $2^3 = 8$.
- Choosing between the biased horses from before (H=2) is $2^2 = 4$.

Relative Entropy

- In real life, we cannot know for sure the exact winning probability for each horse.
- Let's say q is the estimate and p is the true probability (say we got q by observing previous races with these horses)
- We define the *distance* between p and q as the **relative entropy**: written as $D(p\|q)$

$$D(p\|q) = -\sum_{x \in \mathcal{E}} p(x) \log_2 \frac{q(x)}{p(x)}$$

- Note that

$$D(p\|q) = -E_{p(x)}\left[\log_2 \frac{q(x)}{p(x)}\right]$$

- The relative entropy is also called the *Kullback-Leibler divergence*.

Cross Entropy and Relative Entropy

- The **relative entropy** can be written as the sum of two terms:

$$\begin{aligned} D(p\|q) &= -\sum_{x\in\mathcal{E}} p(x)\log_2 \frac{q(x)}{p(x)} \\ &= -\sum_x p(x)\log_2 q(x) + \sum_x p(x)\log_2 p(x) \end{aligned}$$

- We know that $H(p) = -\sum_x p(x)\log_2 p(x)$
- Similarly define $H(p,q) = -\sum_x p(x)\log_2 q(x)$

$$\begin{aligned} D(p\|q) &= H(p,q) & -H(q) \\ \textbf{relative entropy}(p,q) &= \textbf{cross entropy}(p,q) & -\textbf{entropy}(p) \end{aligned}$$

- The term $H(p,q)$ is called the **cross entropy**.

Cross Entropy and Relative Entropy

- $H(p,q) \geq H(p)$ always.
- $D(p\|q) \geq 0$ always, and $D(p\|q) = 0$ iff $p = q$
- $D(p\|q)$ is not a true distance:
  - It is asymmetric: $D(p\|q) \neq D(q\|p)$,
  - It does not obey the triangle inequality: $D(p\|q) \not\leq D(p\|r) + D(r\|q)$

Conditional Entropy and Mutual Information

- *Entropy* of a random variable X:

$$H(X) = -\sum_{x \in \mathcal{E}} p(x) \log_2 p(x)$$

- *Conditional Entropy* between two random variables X and Y:

$$H(X \mid Y) = -\sum_{x,y \in \mathcal{E}} p(x,y) \log_2 p(x \mid y)$$

- *Mutual Information* between two random variables X and Y:

$$I(X;Y) = D(p(x,y) \| p(x)p(y)) = \sum_x \sum_y p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

Log Probability Arithmetic

```
// In Python: use numpy.logaddexp2(x1,x2) for base 2
// computes log(a + b) given log(a) and log(b)
double logadd(double lna, double lnb)
{
    if (lna == 1.0) return lnb;
    if (lnb == 1.0) return lna;
    double diff = lna - lnb;
    // 500 is log of large constant
    if (diff < 500.0)
        return log1p(exp(diff)) + lnb;
        // log1p(x) computes log(1+x)
    else
        return lna;
}
```

Acknowledgements

Many slides borrowed or inspired from lecture notes by Michael Collins, Chris Dyer, Kevin Knight, Philipp Koehn, Adam Lopez, Graham Neubig and Luke Zettlemoyer from their NLP course materials.

All mistakes are my own.

A big thank you to all the students who read through these notes and helped me improve them.