CS463 – Natural Language Processing

Language Model N-gram

- **Word Prediction**
- **Statistical Inference**
  - **Probability Theory**
  - **Conditional Probability**
  - **Bayes' Theorem**
  - **Chain Rule of Probability**
  - **Markov Assumption**
- **N-gram Language Models**
  - **N-grams**
  - **Evaluating Language Models**
  - **Generalization and Zeros**
  - **Smoothing**

Word Prediction

- The quiz was ------
- In this course, I want to get a good -----
- Can I make a telephone -----
- My friend has a fast -----
- This is too -------
- الوقت كالسيف إن لم تقطعه -------
- لا إله إلا أنت سبحانك إني كنت من -------

Word Prediction

- Humans have the ability to predict future words in an utterance.
- How?
  - Domain knowledge
  - Syntactic knowledge
  - Lexical knowledge

Word Prediction

- A useful part of the knowledge is needed to allow Word Prediction (guessing the next word).
  - Start looking at words in context.
  - predict next words in a sequence.
- Word Prediction can be captured using simple statistical techniques.
  - In particular, we'll rely on the notion of the **probability** of a sequence (e.g., sentence) and the **likelihood** of words co-occurring.
- Why word prediction?
  - Why would you want to assign a probability to a sentence? or
  - Why would you want to predict the next word?

Word Prediction

- Many applications employ language models for Word Prediction.
- Examples:
  - Speech recognition
  - Handwriting recognition
  - Spelling correction
  - Machine translation
  - Optical character recognition
  - Augmentative communication

Word Prediction – Application Example

- Word Prediction helps in real world spelling errors:
  - Mental confusions (cognitive)
    - their/they're/there
    - to/too/two
    - weather/whether
  - Typos

Phrases/sentences with errors	**Prediction**
lave for have	*lave: lave, leave or love,* *have: having or shave*
They are leaving in about fifteen minuets to go to her horse.	*horse: house, minuets: minutes*
The study was conducted mainly be John Black.	*be: by*
The design an construction of the system will take …	*an: and*
Hopefully, all with continue smoothly in my absence.	*with: will*
I need to notified the bank of….	*notified: notify*
He is trying to fine out.	*fine: find*

Word Prediction – Application Example

- Word Prediction solution to real world spelling errors:
  1. Collect a set of common pairs of confusions;
  2. Whenever a member of this set is encountered, compute the probability of the sentence in which it appears;
  3. Substitute the other possibilities and compute the probability of the resulting sentence;
  4. Choose the higher one.

Statistical Inference

- Statistical NLP aims to do statistical inference for the field of NL.
- **Statistical inference** consists of taking some data (generated in accordance with some unknown *probability distribution*) and then making some inference about this distribution.
- An example of statistical inference is the task of *language modeling* (ex. how to predict the next word given the previous words)
- In order to do this, we need a *model* of the language.
- Probability theory helps us finding such model

Probability Theory

- How likely it is that an *A* Event (something) will happen.
- **Sample space Ω** is listing of all possible outcome of an experiment.
- Event *A* is a subset of Ω
- Probability function (or distribution)

$$P : \Omega \rightarrow [0,1]$$

- **Prior (unconditional) probability** is the probability before we consider any additional knowledge

$$P(A)$$

Conditional Probability

- Sometimes we have partial knowledge about the outcome of an experiment.
- In such cases **Conditional Probability** applies.
  - Suppose we know that event B is true
  - The probability that event A is true given the knowledge about B is expressed by

$$P(A \mid B)$$

Conditional Probability

- Conditionals

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

- Rearranging

$$P(A \wedge B) = P(A \mid B)P(B)$$

- And also

$$P(A \wedge B) = P(B \mid A)P(A)$$

$$P(A \wedge B) = P(B \wedge A) = P(B \mid A)P(A)$$

Conditional Probability

- Joint probability of *A* and *B*

$$
\begin{aligned}
P(A, B) &= P(A \mid B)P(B) \\
&= P(B \mid A)P(A)
\end{aligned}
$$

Bayes' Theorem

- **Bayes' Theorem** lets us swap the order of dependence between events.
- From Conditional Probability, we saw that

$$P(A|B)=\frac{P(A,B)}{P(B)}$$

- Bayes' Theorem:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem

- We know …

$$P(A \wedge B) = P(A \mid B)P(B)$$

and

$$P(A \wedge B) = P(B \mid A)P(A)$$

- So, rearranging things …

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

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Bayes' Theorem

Bayes' Theorem - Example

- S:stiff neck, M: meningitis
- P(S|M) =0.5 P(M) = 1/50,000 P(S)=1/20
- Someone has stiff neck, should he/she worry?
- Estimate the probability, applying Bayes' Theorem:

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)}$$

$$= \frac{0.5 \times 1/50{,}000}{1/20} = 0.0002$$

Chain Rule of Probability

- The probability of a sequence can be viewed as the probability of a conjunctive event.
- For example, the probability of *"the clever student"* is:

$$P(the \wedge clever \wedge student)$$

Chain Rule of Probability - Example

- Based on Conditional Probability:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

$$P(A \wedge B) = P(A \mid B)P(B)$$
and
$$P(A \wedge B) = P(B \mid A)P(A)$$

$$P(A \wedge B) = P(B \mid A)P(A)$$

- Estimating the probability of the conjunctive event: "the student studies"
  - "the student"

$$P(The \wedge student) = P(student \mid the)P(the)$$

  - "the student studies"

$$P(The \wedge student \wedge studies) =$$
$$P(The)P(student \mid The)P(studies \mid The \wedge student)$$

Chain Rule of Probability

- The probability of a word sequence is the probability of a conjunctive event.

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- The chain rule shows the link between computing the **joint probability of a sequence** and computing the **conditional probability of a word given previous words**.
- Unfortunately, Chain Rule doesn't seem to be really helpful. Why?
  - We don't know how to compute the exact probability of a word given a long sequence of preceding words.
  - Language is creative and any particular context might have never occurred before!

Markov Assumption

- **Markov models** are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past.
- Thus, the **Order of a Markov model** is the length of immediate prior context.
- The assumption that the probability of a word depends only on the previous word is called a **Markov assumption**.

N-gram Language Models

- **Language Models (LMs)** are models that assign probabilities to sequences of words.
- An **n-gram** is a sequence of words:
  - A 2-gram (or **bigram**) is a two-word sequence of words
    - like “please turn”, “turn your”, or ”your homework”.
  - A 3-gram (or **trigram**) is a three-word sequence of words
    - like “please turn your”, or “turn your homework”.
- We use **n-gram models** to estimate the probability of the last word of an n-gram given the previous words, and also to assign probabilities to entire sequences (probability distribution).

N-grams

- A simple **N-gram model** computes $P(w \mid h)$, the probability of a word w given some history h.
  - It uses the previous N-1 words to predict the next one:

    $P(w_n \mid w_{n-1})$

    - Dealing with P(<word> | <some prefix>)
    - unigrams: $P(student)$
    - bigrams: $P(student \mid honest)$
    - trigrams: $P(student \mid clever\ honest)$
    - quadrigrams: $P(student \mid the\ clever\ honest)$

N-grams

- Given a word sequence: $w_1\ w_2\ w_3\ ...\ w_n$
- **Chain rule**
  - $p(w_1\ w_2) = p(w_1)\ p(w_2|w_1)$
  - $p(w_1\ w_2\ w_3) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)$
  - $p(w_1\ w_2\ w_3 ... w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ p(w_4|w_1 w_2 w_3)\ ...\ p(w_n|w_1 ... w_{n-1})$
- Note:
  - It's not easy to collect (meaningful) statistics on $p(w_n|w_{n-1} w_{n-2} ... w_1)$ for all possible word sequences
- **Bigram approximation**
  - *just look at the **previous word only*** (*not all the proceedings words*)
  - Markov Assumption: finite length history
  - 1st order Markov Model
  - $p(w_1\ w_2\ w_3 .. w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ ... p(w_n|w_1 ... w_{n-3} w_{n-2} w_{n-1})$
  - $p(w_1\ w_2\ w_3 .. w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_2) ... p(w_n|w_{n-1})$
- Note:
  - $p(w_n|w_{n-1})$ is a lot easier to estimate well than $p(w_n|w_1 ... w_{n-1})$

N-grams

- Given a word sequence: $w_1\ w_2\ w_3\ ...\ w_n$
- **Chain rule**
  - $p(w_1\ w_2) = p(w_1)\ p(w_2|w_1)$
  - $p(w_1\ w_2\ w_3) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)$
  - $p(w_1\ w_2\ w_3 ... w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ p(w_4|w_1 w_2 w_3)\ ...\ p(w_n|w_1 ... w_{n-1})$
- Note:
  - It's not easy to collect (meaningful) statistics on $p(w_n|w_{n-1} w_{n-2} ... w_1)$ for all possible word sequences
- **Trigram approximation**
  - *just look at the **previous two words only** (not all the proceedings words)*
  - 2nd order Markov Model
  - $p(w_1\ w_2\ w_3\ w_4 ... w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ p(w_4|w_1 w_2 w_3) ... p(w_n|w_1 ... w_{n-3} w_{n-2} w_{n-1})$
  - $p(w_1\ w_2\ w_3 ... w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) p(w_4|w_2 w_3) ... p(w_n\ |w_{n-2}\ w_{n-1})$
- Note:
  - $p(w_n|w_{n-2} w_{n-1})$ is a lot easier to estimate well than $p(w_n|w_1 ... w_{n-1})$ but harder than $p(w_n|w_{n-1})$

N-grams

- Based on Markov assumption, the general equation for n-gram approximation to the conditional probability of the next word in a sequence is

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

- So for each component in the product replace each with its approximation (assuming a prefix (Previous words) of N)
- For a bigram grammar
  - *P(sentence)* can be approximated by multiplying all the bigram probabilities in the sequence
    - *P(I want to eat Chinese food) = P(I | <start>) P(want | I) P(to | want) P(eat | to) P(Chinese | eat) P(food | Chinese) P(<end> | food)*

N-grams

- How do we estimate the bigram or n-gram probabilities?
- To estimate probabilities, we use a method called **Maximum Likelihood Estimation** or **MLE**.
  - Counting from corpus and normalizing the counts so that they lie between 0 and 1

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

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

N-grams

Dan Jurafsky

An example

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

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

N-grams - BErkeley Resturant Project (speech) Example

- BERP bigram counts:

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

N-grams - BErkeley Resturant Project (speech) Example

- Normalization: divide each row's counts by appropriate unigram counts

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

- Computing the probability of I I
  - P = C(I | I) / C(all I)
  - P = 8 / 3437 = .0023
- A bigram grammar is an NxN matrix of probabilities, where N is the vocabulary size

N-grams - BErkeley Resturant Project (speech) Example

$w_{n-1}w_n$ bigram frequencies

W_n

W_{n-1}

	I	want	to	eat	Chinese	food	lunc
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Figure 6.4 Bigram counts for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

I	3437
want	1215
to	3256
eat	938
Chinese	213
food	1506
lunch	459

unigram frequencies

bigram probabilities

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Figure 6.5 Bigram probabilities for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

sparse matrix

zeros probabilities unusable

(*we'll need to do* ***smoothing***)

N-grams - BErkeley Resturant Project (speech) Example

- A Bigram Grammar Fragment from BERP

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

<start> I	.25	Want some	.04
<start> I'd	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

N-grams - BErkeley Resturant Project (speech) Example

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25*.32*.65*.26*.001*.60 = 0.0000081
- P(I want to eat Chinese food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(Chinese|eat) P(food|Chinese) =
  .25*.32*.65*.26*.02*.56 = 0.00015

- What can we infer from these statistics?
- Probabilities seem to capture “syntactic” facts and “world knowledge”
  - eat is often followed by a NP
  - British food is not too popular

N-grams – log probability

- Check the following probabilities:
  - P(I | I) = .0023 I I I I want
  - P(I | want) = .0025 I want I want
  - P(I | food) = .013 the kind of food I want is ...
- Since probabilities are (by definition) less than or equal to 1, the more probabilities we multiply together, the smaller the product becomes.
  - Multiplying enough n-grams together would result in numerical **underflow**.
  - To avoid underflow convert the probabilities to **logs** and then do **additions**.
  - To get the real probability (if you need it) go back to the **antilog**.

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

Evaluating Language Models

- Probabilities come from a training corpus, which is used to design the model.
  - narrow corpus: probabilities don't generalize
  - general corpus: probabilities don't reflect task or domain
- A separate test corpus is used to evaluate the model, typically using standard metrics
  - held out test set
  - cross validation
  - evaluation differences should be statistically significant
    - Try preplexity metric (the inverse probability) to evaluate each model.
    - The lower the preplexity the better the language model.

Evaluating Language Models

- Using **Shannon visualization technique** - choose N-Grams according to their probabilities and string them together to generate random sentences from different n-gram models.
  - Unigrams - Choose a random value between 0 and 1 and print the word whose interval includes this chosen value. We continue choosing random numbers and generating words until we randomly generate the sentence-final token </s>.
  - Bigrams: Start with generating bigrams that start with <s> and has *w* as the second word. We next chose a random bigram starting with *w*, and so on.
    - From BERP:

      <s>I   I want   want to   to eat   eat Chinese   Chinese food   food</s>
- Make sure that the **training** and **testing** datasets share the same **genre** and **dialect**.

Generalization and Zeros

- A small number of events occur with high frequency
  - You can collect reliable statistics on these events with relatively small samples
- A large number of events occur with small frequency
  - You might have to wait a long time to gather statistics on the low frequency events
  - Some zeroes are really zeroes
    - Meaning that they represent events that can’t or shouldn’t occur
  - On the other hand, some zeroes aren’t really zeroes
    - They represent low frequency events that simply didn’t occur in the corpus

Generalization and Zeros

- Problem:
  - Let’s assume we’re using N-grams.
  - How can we assign a probability to a sequence where one of the component n-grams has a value of zero?
    - i.e. words that could be in our vocabulary, but appear in a test set in an unseen context (for example they appear after a word they never appeared after in training)
- Solution - Assume all the words are known and have been seen.
  - Go to a lower order n-gram
  - Back off from bigrams to unigrams
  - Replace the zero with something else

Smoothing

- The simplest way to do smoothing is to add one to all the bigram counts, before we normalize them into probabilities.
  - All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on.
  - Justification: They're just events you haven't seen yet. If you had seen them you would only have seen them once. so make the count equal to 1.
- This algorithm is called **Laplace** smoothing (or **add-one** smoothing).
  - There are other smoothing algorithms too: **Add-k** smoothing, **Backoff** smoothing and **Kneser-Ney** smoothing, but we focus on **Laplace** smoothing.

Unigram: $P(w_i) = \frac{c_i}{N}$ | $P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$

Bigram: $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$ | $P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$

Smoothing – Add-one Smoothing Example (PERP)

- Unsmoothed bigram **counts**:

2nd word

1st word

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N)**
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459
…									

- Unsmoothed bigram **probabilities**:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total**
I	.0023 (8/3437)	.32	0	.0038 (13/3437)	0	0	0		1
want	.0025	0	.65	0	.0049	.0066	.0049		1
to	.00092	0	.0031	.26	.00092	0	.0037		1
eat	0	0	.0021	0	.020	.0021	.055		1
Chinese	.0094	0	0	0	0	.56	.0047		1
food	.013	0	.011	0	0	0	0		1
lunch	.0087	0	0	0	0	.0022	0		1
…									

Smoothing – Add-one Smoothing Example (PERP)

- Add-one smoothed bigram **counts**:

	I	want	to	eat	Chinese	food	lunch	…	**Total (N+V)**
I	~~8~~ 9	~~1087~~ 1088	1	14	1	1	1		~~3437~~ 5053
want	~~3~~ 4	1	787	1	7	9	7		2831
to	4	1	11	861	4	1	13		4872
eat	1	1	23	1	20	3	53		2554
Chinese	3	1	1	1	1	121	2		1829
food	20	1	18	1	1	1	1		3122
lunch	5	1	1	1	1	2	1		2075

- Add-one smoothed bigram **probabilities**:

	I	want	to	eat	Chinese	food	lunch	…	**Total**
I	.0018 (9/5053)	.22	.0002	.0028 (14/5053)	.0002	.0002	.0002		1
want	.0014	.00035	.28	.00035	.0025	.0032	.0025		1
to	.00082	.00021	.0023	.18	.00082	.00021	.0027		1
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021		1
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011		1
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032		1
lunch	.0024	.00048	.00048	.00048	.00048	.0022	.00048		1

Smoothing – Add-one Smoothing Example (PERP)

unsmoothed bigram counts:

$\overbrace{\hphantom{xxxxxxxxxxxxxxxxxxxx}}^{V= 1616 \text{ word types}}$

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N)**
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459

V= 1616

Smoothed P(I eat)

= (C(I eat) + 1) / (number of bigrams starting with "I" + number of possible bigrams starting with "I")

= (13 + 1) / (3437 + 1616)
= 0.0028

Smoothing – Exercise

<s> I am a human </s>

<s> I am not a machine </s>

<s> I I live in KSA </s>

- What is the probability of having the sentence: I am a human?
  - $P(\text{I am a human}) = P(\text{I} \mid \langle s \rangle) * P(\text{am} \mid \text{I}) * P(\text{a} \mid \text{am}) * P(\text{human} \mid \text{a})$

    $= 3/3 * 2/4 * 1/2 * 1/2$

    $= 1 * 0.5 * 0.5 * 0.5 = 0.125$
- What is the probability of having the sentence: I am human?
  - $P(\text{I am human}) = P(\text{I} \mid \langle s \rangle) * P(\text{am} \mid \text{I}) * P(\text{human} \mid \text{am})$

    $= 3/3 * 2/4 * 0/2$

    $= 1 * 0.5 * 0 = 0$

Smoothing – Exercise cont.

<s> I am a human </s>

<s> I am not a machine </s>

<s> I I live in KSA </s>

- General Bigram probability: P(X | Y) = C(XY) / C(Y)
  - P(I am human) = P(I | <s>) * P(am | I) * P(human | am)

    = 3/3 * 2/4 * 0/2

    = 1 * 0.5 * 0 = 0
- Bigram probability with Laplace smoothing:

  P(X|Y) = C(XY)+1 / C(Y)+V
  - P(I am human) = P(I | <s>) * P(am | I) * P(human | am)

    = (3+1) / (3+1) * (2+1) / (4+3) * (0+1) / (2+2)

    = 4/4 * 3/7 * 1/4 = 1 * 0.43 * 0.25 = 0.108