Language Models

Prof. Srijan Kumar
with Roshan Pati and Arindum Roy
Language Models

• What are language models?

 Statistical language models
  – Unigram, bigram and n-gram language model

• Neural language models
Language Models: Objective

• Key question: How well does a model represent the language?
  – **Character language model:** Given alphabet vocabulary $V$, models the probability of generating strings in the language.
  – **Word language model:** Given word vocabulary $V$, models the probability of generating sentences in the language.
Language Model: Applications

- **Assign a probability to sentences**
  - **Machine translation:**
    - \( P(\text{high wind tonight}) > P(\text{large wind tonight}) \)
  - **Spell correction:**
    - The office is about fifteen **minuets** from my house
    - \( P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from}) \)
  - **Speech recognition:**
    - \( P(\text{I saw a van}) \gg P(\text{eyes awe of an}) \)
  - **Information retrieval:** use words that you expect to find in matching documents as your query
  - Many more: **Summarization, question-answering**, and more
Language Models

- What are language models?
  - Statistical language models
  - Neural language models
Language Model: Definition

- **Goal:** Compute the probability of a sentence or sequence of words \( P(s) = P(w_1, w_2, \ldots w_n) \)
- **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 is a language model
- How to compute the joint probability?
  - **Intuition:** apply the chain rule
How To Compute Sentence Probability?

- Given sentence $s = t_1 t_2 t_3 t_4$
- **Applying the chain rule under language model $M$**

$P(t_1 t_2 t_3 t_4 | M) = P(t_1 | M) P(t_2 | M, t_1) P(t_3 | M, t_1 t_2) P(t_4 | M, t_1 t_2 t_3)$

\[
P(\text{○ ○ ○ ●} | M) = P(\text{●} | M)
\]

\[
P(\text{○} | M, \text{●})
\]

\[
P(\text{○} | M, \text{○ ●})
\]

\[
P(\text{●} | M, \text{○ ○ ●})
\]
Complexity of Language Models

• The complexity of language models depends on the window of the word-word or character-character dependency they can handle

• Common types are:
  – **Unigram** language model
  – **Bigram** language model
  – **N-gram** language model
Unigram Model

- **Unigram language model only models the probability of each word according to the model**
  - Does **NOT** model word-word dependency
  - The word order is irrelevant
  - Akin to the “bag of words” model

\[ P(t_1 t_2 t_3 t_4 | \mathcal{M}) = P(t_1 | \mathcal{M}) P(t_2 | \mathcal{M}) P(t_3 | \mathcal{M}) P(t_4 | \mathcal{M}) \]
Bigram Model

- **Bigram language model models the consecutive word dependency**
  - Does **NOT** model longer dependency
  - Word order is relevant here

\[
P(t_1 t_2 t_3 t_4 | M) = P(t_1 | M) P(t_2 | M, t_1) P(t_3 | M, t_2) P(t_4 | M, t_3)
\]
N-gram Model

- **Bigram language model** models the longer sequences of word dependency
  - Most complex among all three

\[ P(t_1 t_2 t_3 t_4 | \mathcal{M}) = P(t_1 | \mathcal{M}) P(t_2 | \mathcal{M}, t_1) P(t_3 | \mathcal{M}, t_1 t_2) P(t_4 | \mathcal{M}, t_1 t_2 t_3) \]
Unigram Language Model: Example

- **What is the probability of the sentence s under language model M?**
- **Example:**
  
  "the man likes the woman"
  
  \[
  P(s \mid M) = 0.2 \times 0.01 \times 0.02 \times 0.2 \times 0.01 = 0.00000008
  \]

<table>
<thead>
<tr>
<th>Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
</tr>
<tr>
<td>man</td>
<td>0.01</td>
</tr>
<tr>
<td>woman</td>
<td>0.01</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Language Model M
Comparing Language Models

• **Given two language models, how can we decide which language model is better?**

• **Solution:**
  – Take a set $S$ of sentences we desire to model
  – For each language model:
    • Find the probability of each sentence
    • Average the probability scores
  – The language model with the highest average probability is the best fit for language model
Comparing Language Models

- **s**: “the man likes the woman”
- **M1**: $0.2 \times 0.01 \times 0.02 \times 0.2 \times 0.01 \Rightarrow P(s|M1) = 0.00000008$
- **M2**: $0.1 \times 0.1 \times 0.01 \times 0.1 \times 0.1 \Rightarrow P(s|M2) = 0.0000001$
- $P(s|M2) > P(s|M1) \Rightarrow M2$ is a better language model

<table>
<thead>
<tr>
<th>Word</th>
<th>Probability</th>
<th>Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>the</td>
<td>0.1</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>a</td>
<td>0.02</td>
</tr>
<tr>
<td>man</td>
<td>0.01</td>
<td>man</td>
<td>0.1</td>
</tr>
<tr>
<td>woman</td>
<td>0.01</td>
<td>woman</td>
<td>0.1</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>said</td>
<td>0.02</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>likes</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Estimating Probabilities

- N-gram conditional probability can be estimated based on the raw occurrence counts in the observed corpus

- Uni-gram
  \[ P(t_n) = C(t_n) \]

- Bi-gram
  \[ P(t_n | t_{n-1}) = \frac{C(t_{n-1} t_n)}{C(t_{n-1})} \]

- N-gram
  \[ P(t_n | t_{n-N-1:n-1}) = \frac{C(t_{n-N-1} \cdots t_{n-1} t_n)}{C(t_{n-N-1} \cdots t_{n-1})} \]
Estimating Bigram Probabilities: Case Study

• Corpus: 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: Case Study

- **Bigram matrix created from 9222 sentences**

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Raw Bigram Probabilities: Case Study

- **Unigram counts**

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

- **Normalize by unigrams**

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00079</td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0</td>
<td>0.0054</td>
<td>0.0011</td>
</tr>
<tr>
<td>to</td>
<td>0.00083</td>
<td>0</td>
<td>0.0017</td>
<td>0.28</td>
<td>0.00083</td>
<td>0</td>
<td>0.0025</td>
<td>0.087</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0.0027</td>
<td>0</td>
<td>0.0021</td>
<td>0.52</td>
<td>0.0037</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0063</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00092</td>
<td>0.0029</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0.014</td>
<td>0</td>
<td>0.14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0059</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>0.0036</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ P(\text{want} \mid i) = \frac{C(i, \text{want})}{C(i)} = \frac{827}{2533} = 0.33 \]
Language Models

- What are language models?
- Statistical language models
  - Unigram, bigram, and n-gram language models
- Neural language models
- Language models for IR
Neural Language Models

• So far, the language models have been statistics and counting based
• Now, language models are created using neural networks/deep learning
• Key question: how to model sequences?
Neural-based Bigram Language Mode

Problem: Does not model sequential information (too local)
Sequences in Inputs or Outputs?

**Input:** No sequence

**Output:** No seq.

**Example:**
“standard”
classification / regression problems
Sequences in Inputs or Outputs?

**Input**: No sequence  
**Output**: No seq.  
**Example**: “standard” classification / regression problems

**Input**: No sequence  
**Output**: No seq.  
**Example**: Im2Caption

**Input**: Sequence  
**Output**: Sequence  
**Example**: machine translation, video captioning, open-ended question answering, video question answering

**Input**: Sequence  
**Output**: No seq.  
**Example**: sentence classification, multiple-choice question answering

**Input**: No sequence  
**Output**: No seq.  
**Example**: sentence classification, multiple-choice question answering
Key Conceptual Ideas

• **Parameter Sharing**
  – in computational graphs = adding gradients
• **“Unrolling”**
  – in computational graphs with parameter sharing
• **Parameter Sharing + “Unrolling”**
  – Allows modeling *arbitrary length sequences*!
  – Keeps number of parameters in check
Recurrent Neural Network

usually want to predict a vector at some time steps
Recurrent Neural Network

- We can process a sequence of vectors $x$ by applying a recurrence formula at every time step.
- $f_W$ is used at every time step and shared across all data.

\[
\begin{align*}
    h_t &= f_W(h_{t-1}, x_t) \\
    \text{new state} &\quad \text{old state} \\
    \text{some function with parameters } W &\quad \text{input vector at some time step}
\end{align*}
\]
(Vanilla) Recurrent Neural Network

\[ h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \]

\[ h_t = f_W(h_{t-1}, x_t) \]

\[ y_t = W_{hy}h_t + b_y \]
RNN Computational Graph

Initial hidden state

\[ h_0 \rightarrow f_W \rightarrow h_1 \rightarrow f_W \rightarrow h_2 \rightarrow f_W \rightarrow h_3 \rightarrow \cdots \rightarrow h_T \]

Input at time 1
\[ X_1 \]

Input at time 2
\[ X_2 \]

Input at time 3
\[ X_3 \]

Final hidden state
RNN Computational Graph

- The same weight matrices $W$ is shared for all time steps
RNN Computational Graph: Many to Many

- Many-to-many architecture has one output per time step

\[ h_0 \rightarrow f_W \rightarrow h_1 \rightarrow f_W \rightarrow h_2 \rightarrow f_W \rightarrow h_3 \rightarrow \cdots \rightarrow h_T \]

Output at time 1
Output at time 2
Output at time 3
Final output

\[ y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow y_T \]
RNN Computational Graph: Many to Many
RNN Computational Graph: Many to Many

Total loss
RNN Computational Graph: Many to one

- Many-to-one architecture has one final output
RNN Computational Graph: One to many

- Many-to-one architecture has one input and several outputs
Example: Character-level Language Model

- **Input:** one hot representation of the characters
- **Vocabulary:** [‘h’, ‘e’, ‘l’, ‘o’]

Example training sequence:
“hello”
Example: Character-level Language Model

- Transform every input into the hidden vector

\[ h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \]

Example training sequence: “hello”
Example: Character-level Language Model

- Transform each hidden vector into a output vector

Example training sequence: “hello”
Example: Generating Output via Sampling

- **Generating output:** Sample from the vocabulary based on the normalized output layer.
- At test time, sample one character at a time and feed back into the RNN model at the next time step.
Calculating Loss: BackProp Through Time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient
Truncated BackProp Through Time

Run backwards and forwards through (fixed length) **chunks of the sequence**, instead of the whole sequence
Truncated BackProp Through Time

Run backwards and forwards through (fixed length) **chunks of the sequence**, instead of the whole sequence.

Carry hidden states forward, but only BackProp through some smaller number of steps.
Truncated BackProp Through Time

Run backwards and forwards through (fixed length) **chunks of the sequence**, instead of the whole sequence.
Example: Learning to Write Like Shakespeare

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'th thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou art now the world's fresh ornament,
And only herald to the gaudy spring.
Within thine own bud buried thy content,
And tender churl mak'st waste in niggarding:

Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!

'This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.'
Example: Learning to Write Like Shakespeare

at first:

tyntd-iafhatawiaohrdemot lytdws e ,ftti, astai f ogoh eeose rrranbyne 'nhthnee e plia tklrgrd t o idoe ns,smtt h ne etie h,hregtrs nigtkie,aaoenns lng

↓ train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, amerenith ol sivh I latelthernd Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearily, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overetical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
Example: Learning to Write Like Shakespeare

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain’d into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I’ll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I’ll drink it.

VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
Example: Learning to Code

Trained on entire source code of Linux kernel

```c
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << i))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000ffffffff & 0x0000000f) << 8);
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &offset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```
Example: Generating Rap Lyrics

```
Everybody got one
And all the pretty mommies want some
And what i told you all was
But you need to stay such do not touch
They really do not want you to vote
what do you condone
Music make you lose control
What you need is right here ahh oh
This is for you and me
I had to dedicate this song to you Mami
Now I see how you can be
I see u smiling i kno u hattig
Best I Eva Had x4
That I had to pay for
Do I have the right to take yours
Trying to stay warm
```

(2 Chainz - Extremely Blessed)
(Mos Def - Undeniable)
(Lil Wayne - Welcome Back)
(Common - Heidi Hoe)
(KRS One - The Mind)
(Cam'ron - Bubble Music)
(Missy Elliot - Lose Control)
(Wiz Khalifa - Right Here)
(Missy Elliot - Hit Em Wit Da Hee)
(Fat Joe - Bendicion Mami)
(Lil Wayne - How To Hate)
(Wiz Khalifa - Damn Thing)
(Nicki Minaj - Best I Ever Had)
(Ice Cube - X Bitches)
(Common - Retrospect For Life)
(Everlast - 2 Pieces Of Drama)
Example: Movie generated by AI

Sunspring, a short science fiction movie written entirely by AI, debuts exclusively on Ars today.
Complex RNNs: Multilayer

- **Multilayer RNNs**: Create multiple layers of hidden layers on top of one another.

\[ h_t^l = \tanh W^l \left( h_{t-1}^l \right) \]

\[ h \in \mathbb{R}^n. \quad W^l \ [n \times 2n] \]
Long-Short Term Memory (LSTM)

- **Problem with RNN:** can not model long sequences well
  - **Vanishing gradient problem:** gradient of the loss function decays exponentially with time
- **LSTM overcomes this issue**

**Vanilla RNN**

\[
h_t = \tanh \left( W \left( h_{t-1}, x_t \right) \right)
\]

**LSTM**

\[
\begin{pmatrix}
i \\ f \\ o \\ g
\end{pmatrix} = \begin{pmatrix}
\sigma \\ \sigma \\ \sigma \\ \sigma
\end{pmatrix} W \begin{pmatrix}
h_{t-1} \\ x_t
\end{pmatrix}
\]
\[
c_t = f \odot c_{t-1} + i \odot g
\]
\[
h_t = o \odot \tanh(c_t)
\]
LSTM Architecture
Long-Short Term Memory (LSTM)

- **Cell State**: long-term memory of the information
LSTM Intuition: Forget Gate

- **Forget gate**: should we remember the past information?

\[
f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right)
\]

**Intuition**: memory and forget gate output multiply, output of forget gate can be though of as binary (0 or 1)

- anything x 1 = anything (remember)
- anything x 0 = 0 (forget)
LSTM Intuition: Input Gate

- **Input gate:** should we update the memory using the new information bit? If so, by how much?

\[
i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\]
LSTM Intuition: Memory Update

• Forget what needs to be forgotten, update what needs to be updated

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]
LSTM Intuition: Output Gate

- Should we output this bit of information, i.e., to deeper LSTM layers?

\[
o_t = \sigma (W_o \ [h_{t-1}, x_t] + b_o)
\]

\[
h_t = o_t \cdot \tanh (C_t)
\]
LSTM Intuition: Additive Updates

- **Backpropagation** from $c_t$ to $c_{t-1}$ requires **only** elementwise multiplication by $f$, no matrix multiply by $W$
LSTM Intuition: Additive Updates

Uninterrupted gradient flow!
Gated Recurrent Unit (GRU)

- Simpler than LSTM
- No separate memory unit, memory = hidden output

\[
z_t = \sigma (W_z \cdot [h_{t-1}, x_t])
\]
\[
r_t = \sigma (W_r \cdot [h_{t-1}, x_t])
\]
\[
\tilde{h}_t = \tanh (W \cdot [r_t \cdot h_{t-1}, x_t])
\]
\[
h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t
\]

\(z = \text{memorize new and forget old}\)
Language Models

- What are language models?
  - Statistical language models
    - Unigram, bigram, and n-gram language models
  - Neural language models
- Language models for IR

Parts of this lecture are inspired by ChengXiang Zhai
Probability of Relevance

- Three random variables: Query Q, Document D, Relevance R ∈ \{0, 1\}
- **Key question:** what is the probability that THIS document is relevant to THIS query?
- **Goal:** Given a particular query q, a particular document d, 
  \( p(R=1|Q=q,D=d)=? \)
  - Then, rank D based on \( P(R=1|Q,D) \)
- **Solution:** Language modeling based
Defining \( P(R=1|Q,D) \): Generative models

**Basic idea**
- Define \( P(Q,D|R) \)
- Compute \( O(R=1|Q,D) \) using **Bayes’ rule**

\[
O(R = 1 | Q, D) = \frac{P(R = 1 | Q, D)}{P(R = 0 | Q, D)} = \frac{P(Q, D | R = 1)}{P(Q, D | R = 0)} \frac{P(R = 1)}{P(R = 0)}
\]

**Types of models**

- **Query “generation” model**: \( P(Q,D|R) = P(Q|D,R)P(D|R) \)
- **Document “generation” model**: \( P(Q,D|R) = P(D|Q,R)P(Q|R) \)

Ignored for ranking \( D \)
Query Generation: A Language Models for IR

\[ O(R = 1 | Q, D) \propto \frac{P(Q, D | R = 1)}{P(Q, D | R = 0)} \]

\[ = \frac{P(Q | D, R = 1)P(D | R = 1)}{P(Q | D, R = 0)P(D | R = 0)} \]

\[ \propto P(Q | D, R = 1) \frac{P(D | R = 1)}{P(D | R = 0)} \]

(Assume \( P(Q | D, R = 0) \approx P(Q | R = 0) \))

Query likelihood \( p(Q|D,R=1) \) Document prior

• Assuming uniform prior, we have \( O(R = 1 | Q, D) \propto P(Q | D, R = 1) \)

• \( P(Q|D, R=1) = \) Probability that a user who likes \( D \) would pose query \( Q \).

• How to estimate it?
Estimating Probabilities

• How to compute $P(Q|D, R=1)$?
• The Basic LM Approach, by Ponte & Croft 1998
• **Generally involves two steps:**
  1. Estimate a language model based on $D$
  2. Compute the query likelihood according to the estimated model
Ranking Docs by Query Likelihood

Step 1: Given a document, generate a language model

Step 2: Compute query likelihood from the document language model

\[ \text{Doc LM} \]

\[ d_1 \rightarrow \theta_{d_1} \rightarrow p(q|\theta_{d_1}) \rightarrow q \]

\[ d_2 \rightarrow \theta_{d_2} \rightarrow p(q|\theta_{d_2}) \]

\[ d_N \rightarrow \theta_{d_N} \rightarrow p(q|\theta_{d_N}) \]
Example

Document

Language Model

Query = “data mining algorithms”

Which model would most likely have generated this query?

Text mining paper

Food nutrition paper

... text ?
mining ?
assocation ?
clustering ?
...
food ?
...

...
food ?
nutrition ?
healthy ?
diet ?
...

...
Modeling Queries: Multi-Bernoulli Model

- **Multi-Bernoulli: Modeling word presence/absence**
  - \( q = (x_1, \ldots, x_{|V|}) \)
  - \( x_i = 1 \) for presence of word \( w_i \); \( x_i = 0 \) for absence

\[
p(q = (x_1, \ldots, x_{|V|}) \mid d) = \prod_{i=1}^{|V|} p(w_i = x_i \mid d) = \prod_{i=1, x_i=1}^{|V|} p(w_i = 1 \mid d) \prod_{i=1, x_i=0}^{|V|} p(w_i = 0 \mid d)
\]

- Parameters: \{p(w_i=1|d), p(w_i=0|d)\}
  - \( p(w_i=1|d) + p(w_i=0|d) = 1 \)
Modeling Queries: Multinomial Model

- **Multinomial Language Model: Modeling word frequency**
  
  - \( q = (q_1, \ldots, q_m) \), where \( q_j \) is a query word
  
  \[
  p(q = q_1 \ldots q_m \mid d) = \prod_{j=1}^{m} p(q_j \mid d) = \prod_{i=1}^{V} p(w_i \mid d)^{c(w_i, q)}
  \]
  
  - \( c(w_i, q) \) is the count of word \( w_i \) in query \( q \)
  
  - Parameters: \( \{p(w_i \mid d)\} \)
    
    - \( p(w_1 \mid d) + \ldots + p(w_V \mid d) = 1 \)

- **Multinomial language model has performed better than Bernoulli**
Retrieval as LM Estimation

- **Using Multinomial Language Model**
  \[
p(q = q_1\ldots q_m \mid d) = \prod_{j=1}^{m} p(q_j \mid d) = \prod_{i=1}^{|V|} p(w_i \mid d)^{c(w_i,q)}
\]

- **Document ranking based on query likelihood**
  \[
  \log p(q \mid d) = \sum_{i=1}^{m} \log p(q_i \mid d) = \sum_{i=1}^{|V|} c(w_i,q) \log p(w_i \mid d)
  \]

  where, \( q = q_1q_2\ldots q_m \)

- **Retrieval problem \approx Estimation of \( p(w_i \mid d) \)**
How To Estimate $P(w|d)$?

- **Simplest solution: Maximum Likelihood Estimator**
  - $P(w|d) =$ relative frequency of word $w$ in $d$
- What if a word doesn’t appear in the text? Then $P(w|d)=0$
  - What probability should we give a word that has not been observed?
  - Requires smoothing
How To Smooth A Language Model

- **Key Question:** what probability should be assigned to an unseen word?
- **Solution:** Let the probability of an unseen word be proportional to its probability given by a reference LM.
- **Example:** Reference LM = Collection LM

\[
p(w | d) = \begin{cases} 
p_{\text{seen}}(w | d) & \text{if } w \text{ is seen in } d \\ 
\alpha_d p(w | C) & \text{otherwise} 
\end{cases}
\]

Discounted ML estimate

Collection language model
Rewriting the Ranking Function with Smoothing

\[
\log p(q \mid d) = \sum_{w \in V} c(w, q) \log p(w \mid d)
\]

\[
= \sum_{w \in V, c(w, d) > 0} c(w, q) \log p_{\text{seen}}(w \mid d) + \sum_{w \in V, c(w, d) = 0} c(w, q) \log \alpha_d p(w \mid C)
\]

Query words matched in \(d\)

Query words not matched in \(d\)

\[
= \sum_{w \in V, c(w, d) > 0} c(w, q) \log \frac{p_{\text{seen}}(w \mid d)}{\alpha_d p(w \mid C)} + |q| \log \alpha_d + \sum_{w \in V} c(w, q) \log p(w \mid C)
\]

All query words

Query words matched in \(d\)
Benefit of Rewriting

- Better understanding of the ranking function
  - Smoothing with \( p(w|C) \) \( \Rightarrow \) TF-IDF weighting + length norm.

\[
\log p(q | d) = \sum_{w_i \in d} c(w, q) \left[ \log \frac{p_{\text{Seen}}(w_i | d)}{\alpha_d p(w_i | C)} \right] + n \log \alpha_d + \sum_{i=1}^{n} \log p(w_i | C)
\]

- TF weighting
- Doc length normalization
- IDF weighting
- Ignore for ranking

- Enable efficient computation