Logistics

• Presentation slots

• Lecture videos posted on class channel on Mediaspace

• Assignment 1 out
  • due 2/11
  • post issues on Piazza
  • submit on Gradescope
From Words to Word Sequences

• Words as units of text
  • Word level models for text classification

• Relations between words
  • Word meaning and similarity
Words to Word-Sequences

NLP rich in sequences
- Characters to words
- Words to sentences
- Sentences to documents

- Two models of words as sequences
  - Language modeling
  - Tagging
Words to Word Sequences

• Language modeling

• Tagging
Which of These are Valid?

• Iryna went to the museum.

• museum Iryna to the went.

• Iryna went museum.

• The museum went Iryna.

• The mobile museum went to Iryna.
Language Modeling

• Probability of a sentence (sequence of words)
  • \( p(w_1, w_2, \ldots, w_M) \), with \( w_m \in V \) (vocabulary)

• Why is probability of a sentence useful?
  • Machine translation

他向记者介绍了发言的主要内容
– He briefed to reporters on the chief contents of the statement
– He briefed reporters on the chief contents of the statement
– He briefed to reporters on the main contents of the statement
– He briefed reporters on the main contents of the statement
Language Modeling

• Probability of a sentence (sequence of words)
  • $p(w_1, w_2, \ldots, w_M)$, with $w_m \in V$ (vocabulary)

• Why is probability of a sentence useful?
  • Machine translation
  • Speech recognition
  • Summarization
  • Dialog generation
Language Modeling

- Everyday use of LM
  - Given a part of sentence, predict next word
Language Modeling

- Probability of a sentence
- Measure of fluency of sentence

- El café negro me gusta mucho.

{the coffee black me pleases much, I love black coffee}
N-Gram Language Modeling

• Classical models for LM
  • Definition: \textit{n-gram is a chunk of n consecutive words}
  • Unigram, bigram, trigram

• Core idea:
  • Gather statistics on n-grams from a corpus
  • Use to predict next word/probability of sentence
N-Gram Language Modeling

• Classical models for LM
  • $n$-gram language models
• Distribution of next word is a multinomial conditioned on previous $n-1$ words
  \[
P(W) = P(w_1, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i | w_1, \ldots, w_{i-1})\]

• Simplifying assumption: $k$-th order Markov assumption
  K-gram model condition on $k-1$ words
  \[
P(w_n | w_1, \ldots, w_{n-1}) \approx P(w_n | w_{n-k+1}, \ldots, w_{n-1})\]
  • trigram model $P(w_1, \ldots, w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \ldots$
Estimating Probabilities

\[ P(w|\text{visited San}) = \frac{\text{count}(\text{visited San}, w)}{\text{count}(\text{visited San})} \]

- Assume we have a vocabulary of size \( V \), how many sequences of length \( n \) do we have?
  A) \( n \times V \)
  B) \( n^V \)
  C) \( V^n \)
  D) \( V/n \)
How to Learn a LM?

\[ P(W) = P(w_1, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i | w_{i-k+1} \ldots w_{i-1}) \]

- Conditional probabilities
- Obtained by MLE (counting)

- *I visited San _____*
- put a distribution on next word using trigram language model learned from large corpus

\[ P(w|visited \ San) = \frac{\text{count}(visited \ San, w)}{\text{count}(visited \ San)} \]
How to Learn a LM?

• Pad a \texttt{\textless begin\textgreater} and \texttt{\textless end\textgreater} symbol

• Count to obtain MLE of probabilities

• \(P(\text{I like black coffee}) = P(\text{I}| \text{\textless begin\textgreater}) \ldots P(\text{coffee}|\text{black}).P(\text{\textless end\textgreater}| \text{coffee})\)
Problems with N-gram LM?

• Throwing away too much context, impacts the word we predict

• 4-gram LM
  When the lunch bell rang, the students opened their _______

• When the lunch bell rang, the students opened their _______
Problems with N-gram LM?

• Sparsity issues

\[ P(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})} \]

• For some \( w \), the count of numerator is zero

solution: smoothing, have small probability for every \( w \)
Smoothing

We often want to make estimates from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total

Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total
Problems with N-gram LM?

• Sparsity issues

\[
P(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}
\]

• Sparsity in terms of count of denominator
  • Solution: Back off

• Worsens for large \( n \), so \( n \leq 5 \) typically

• Number of parameters grows with \( n \)
All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, ...

That’s why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.
What else can you use LMs for?

- Generate text
  - <start> I love ____
  - <start> I love to ____

while didn’t choose end-of-sentence symbol:
- calculate probability
- sample a new word from the probability distribution
Evaluating LM

- Extrinsic: check whether the language model improves a task

- Intrinsic: Best LM is one that best predicts an unseen test set
  - Gives the highest $P(\text{sentence})$
Evaluating LM

- Extrinsic: check whether the language model improves a task

- Intrinsic: held-out likelihood on tests

\[ \ell(w) = \sum_{m=1}^{M} \log p(w_m | w_{m-1}, \ldots, w_1), \]

Perplexity: inverse probability of the test set, normalized by the number of words

\[ \text{Perplex}(w) = 2^{-\frac{\ell(w)}{M}}, \]

Minimizing perplexity == maximizing probability
<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to compute</td>
<td>Requires domain match between train and test</td>
</tr>
<tr>
<td>standardized</td>
<td>might not correspond to end task optimization</td>
</tr>
<tr>
<td>directly useful, easy to use to correct sentences</td>
<td>log 0 undefined</td>
</tr>
<tr>
<td>nice theoretical interpretation - matching distributions</td>
<td>can be ‘cheated’ by predicting common tokens</td>
</tr>
<tr>
<td></td>
<td>size of test set matters</td>
</tr>
<tr>
<td></td>
<td>can be sensitive to low prob tokens/sentences</td>
</tr>
</tbody>
</table>
Problems and Solutions

• Cannot share strength among **similar words**
  
  she **bought** a car  
  she **bought** a bicycle  
  she **purchased** a car  
  she **purchased** a bicycle  

  → solution: class based language models

• Cannot condition on context with **intervening words**
  
  Dr. Jane **Smith**  
  Dr. Gertrude **Smith**  

  → solution: skip-gram language models

• Cannot handle **long-distance dependencies**
  
  for **tennis** class he wanted to buy his own **racquet**  
  for **programming** class he wanted to buy his own **computer**

  → solution: cache, trigger, topic, syntactic models, etc.
Alternative: Featurized Linear Models

• Calculate features of the context
• Based on the features, calculate probabilities
• Optimize feature weights using gradient descent
Example

Previous words: “giving a"

\[
b = \begin{pmatrix}
    3.0 \\
    2.5 \\
    -0.2 \\
    0.1 \\
    1.2 \\
    \vdots
\end{pmatrix}, \quad w_{1,a} = \begin{pmatrix}
    -6.0 \\
    -5.1 \\
    0.2 \\
    0.1 \\
    0.5 \\
    \vdots
\end{pmatrix}, \quad w_{2,giving} = \begin{pmatrix}
    -0.2 \\
    -0.3 \\
    1.0 \\
    2.0 \\
    -1.2 \\
    \vdots
\end{pmatrix}, \quad s = \begin{pmatrix}
    -3.2 \\
    -2.9 \\
    1.0 \\
    2.2 \\
    0.6 \\
    \vdots
\end{pmatrix}
\]

Words we’re predicting | How likely are they? | How likely are they given prev. word is “a”? | How likely are they given 2nd prev. word is “giving”? | Total score
---|---|---|---|---

Convert scores into probabilities by taking the exponent and normalizing (softmax)
Problems and Solutions

• Cannot share strength among **similar words**

  | she bought a car          | she bought a bicycle       |
  | she purchased a car       | she purchased a bicycle    |

  → not solved yet 😞

• Cannot condition on context with **intervening words**

  | Dr. Jane Smith           | Dr. Gertrude Smith        |

  → solved! 😊

• Cannot handle **long-distance dependencies**

  | for tennis class he wanted to buy his own racquet |
  | for programming class he wanted to buy his own computer |

  → not solved yet 😞
Linear Models Can’t Learn Feature Combinations

students take tests $\rightarrow$ **high**  
students write tests $\rightarrow$ **low**  
teachers take tests $\rightarrow$ **low**  
teachers write tests $\rightarrow$ **high**

- These can’t be expressed by linear features
- What can we do?
  - Remember combinations as features (individual scores for “students take”, “teachers write”)
    $\rightarrow$ Feature space explosion!
  - Neural nets
Neural Networks

• Complex models for NLP

• Text classification
Text Classification

A First Try:
Bag of Words (BOW)
Text Classification

- Each word has its own 5 elements corresponding to [very good, good, neutral, bad, very bad]
- “hate” will have a high value for “very bad”, etc.

- Does it contain “don’t” and “love”?
- Does it contain “don’t”, “i”, “love”, and “nothing”?

A First Try: Bag of Words (BOW)

I don’t love this movie

There’s nothing I don’t love about this movie
Neural Networks for Text Classification

I  
lookup

hate  
lookup

this  
lookup

movie  
lookup

some complicated function to extract combination features (neural net)

scores

probs

softmax
Continuous Bag of Words (CBOW)

- Still no combination features: only the expressive power of a linear model, but dimension reduced
Deep CBOW

\[ \text{hate} + \text{this} + \text{movie} = \tanh(W_1 \cdot h + b_1) + \tanh(W_2 \cdot h + b_2) \]

\[ W + \text{bias} = \text{scores} \]
Neural Networks for Text Classification

- Now things are more interesting!
- We can learn feature combinations (a node in the second layer might be “feature 1 AND feature 5 are active”)
- e.g. capture things such as “not” AND “hate”
Neural Language Models

- (See Bengio et al. 2004)
Neural Language Models = Shared Strength

Word embeddings: Similar input words get similar vectors

Similar output words get similar rows in the softmax matrix

Similar contexts get similar hidden states

\[
\text{tanh}(W_1^\top h + b_1)
\]

\[
W + \text{bias} \quad \rightarrow \quad \text{scores} \quad \rightarrow \quad \text{softmax} \quad \rightarrow \quad \text{probs}
\]
Problems and Solutions

- Cannot share strength among **similar words**

<table>
<thead>
<tr>
<th>she bought a car</th>
<th>she bought a bicycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>she purchased a car</td>
<td>she purchased a bicycle</td>
</tr>
</tbody>
</table>

  → solved, and similar contexts as well! 😊

- Cannot condition on context with **intervening words**

  | Dr. Jane Smith       | Dr. Gertrude Smith |

  → solved! 😊

- Cannot handle **long-distance dependencies**

  | for tennis class he wanted to buy his own racquet |
  | for programming class he wanted to buy his own computer |

  → not solved yet 😞
Long Range Dependencies

- Agreement in number, gender, etc.
  
  He does not have very much confidence in himself.
  She does not have very much confidence in herself.

- Selectional preference

  The reign has lasted as long as the life of the queen.
  The rain has lasted as long as the life of the clouds.
Long Range Dependencies

- What is the referent of “it”? 

  The trophy would not fit in the brown suitcase because it was too **big**.

  **Trophy**

  The trophy would not fit in the brown suitcase because it was too **small**.

  **Suitcase**

(from Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)
Recurrent Neural Networks (Elman 1990)

Feed-forward NN

- context
- lookup
- transform
- predict
- label

Recurrent NN

- context
- lookup
- transform
- predict
- label
Recurrent Neural Networks (Elman 1990)

- What does processing a sequence look like?
RNN Training

1 → hate → this → movie → prediction 1 → prediction 2 → prediction 3 → prediction 4 → sum → total loss

loss 1 → label 1 → loss 2 → label 2 → loss 3 → label 3 → loss 4 → label 4
RNN Advantage

- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point
Represent Sentences

\[ \text{RNN} \rightarrow \text{RNN} \rightarrow \text{RNN} \rightarrow \text{RNN} \rightarrow \text{predict} \]

\[ \text{I} \rightarrow \text{hate} \rightarrow \text{this} \rightarrow \text{movie} \rightarrow \text{prediction} \]
Represent Sentences

- Sentence classification
- Conditioned generation
- Retrieval
RNN Advantage

- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point
Represent Contexts

I → RNN → predict → label

hate → RNN → predict → label

this → RNN → predict → label

movie → RNN → predict → label
Represent Contexts: Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
Bidirectional RNNs

- A simple extension, run the RNN in both directions