Language Models and Transfer Learning

Yifeng Tao
School of Computer Science
Carnegie Mellon University

Slides adapted from various sources (see reference page)
What is a Language Model

- A statistical language model is a probability distribution over sequences of words.
- Given such a sequence, say of length $m$, it assigns a probability to the whole sequence:
  
  $$P(w_1, \ldots, w_m)$$

- Main problem: data sparsity

[Slide from https://en.wikipedia.org/wiki/Language_model.]
Unigram model: Bag of words

- General probability distribution:

\[ P(t_1 t_2 t_3) = P(t_1)P(t_2 | t_1)P(t_3 | t_1 t_2) \]

- Unigram model assumption:

\[ P_{\text{uni}}(t_1 t_2 t_3) = P(t_1)P(t_2)P(t_3) \]

- Essentially, bag of words model

- Estimation of unigram params: count word frequency in the doc

<table>
<thead>
<tr>
<th>Terms</th>
<th>Probability in doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.1</td>
</tr>
<tr>
<td>world</td>
<td>0.2</td>
</tr>
<tr>
<td>likes</td>
<td>0.05</td>
</tr>
<tr>
<td>we</td>
<td>0.05</td>
</tr>
<tr>
<td>share</td>
<td>0.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Slide from https://en.wikipedia.org/wiki/Language_model.]
n-gram model

○ n-gram assumption:

\[ P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \]

○ Estimation of n-gram params:

\[ P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})} \]

[Slide from https://en.wikipedia.org/wiki/Language_model.]
Word2Vec

- Word2Vec: Learns distributed representations of words
- **Continuous bag-of-words (CBOW)**
  - Predicts current word from a window of surrounding context words
- **Continuous skip-gram**
  - Uses current word to predict surrounding window of context words
  - Slower but does a better job for infrequent words

[Slide from https://www.tensorflow.org/tutorials/representation/word2vec.]
Skip-gram Word2Vec

- All words: $\mathcal{G}$

- Parameters of skip-gram word2vec model
  - Word embedding for each word: $\mathcal{E} = \{ e_g \in \mathbb{R}^n \} \, g \in \mathcal{G}$
  - Context embedding for each word: $\mathcal{V} = \{ v_g \in \mathbb{R}^n \} \, g \in \mathcal{G}$

- Assumption:

$$
Pr \left( c \in \text{Context}(g) \mid g \right) = \frac{\exp \left( e_g^T v_c \right)}{\sum_{c' \in \mathcal{G}} \exp \left( e_g^T v_{c'} \right)}
$$

[Slide from https://www.tensorflow.org/tutorials/representation/word2vec.]
Distributed Representations of Words

- The trained parameters of words in skip-gram word2vec model
- Semantics and embedding space

[Slide from https://www.tensorflow.org/tutorials/representation/word2vec.]
Word Embeddings in Transfer Learning

- **Transfer learning:**
  - Labeled data are limited
  - Unlabeled text corpus enormous
  - Pretrained word embeddings can be transferred to other supervised tasks.
    E.g., POS, NER, QA, MT, Sentiment classification

[Slide from Matt Gormley.]
SOTA Language Models: ELMo

- Embeddings from Language Models: ELMo
  - Fits full conditional probability in forward direction:
    \[
    p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).
    \]
  - Fits full conditional probability in both directions using LSTM:
    \[
    \sum_{k=1}^{N} \left( \log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \Theta_{LSTM}, \Theta_s) \\
    + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \Theta_{LSTM}, \Theta_s) \right)
    \]

SOTA Language Models: OpenAI GPT & BERT

- Uses transformer other than LSTM to model language
  - OpenAI GPT: single direction
  - BERT: bi-direction

[Slide from https://arxiv.org/abs/1810.04805.]
SOTA Language Models: BERT

- Additional language modeling task: predict whether sentences come from same paragraph.

[Slide from https://arxiv.org/abs/1810.04805.]
Instead of extract embeddings and hidden layer outputs, can be fine-tuned to specific supervised learning tasks.

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

[Slide from https://arxiv.org/abs/1810.04805.]
The Transformer and Attention Mechanism

- An encoder-decoder structure
- Our focus: encoder and attention mechanism

[Slide from https://jalammar.github.io/illustrated-transformer/]
The Transformer and Attention Mechanism

- **Self-attention**
  - Ignores positions of words, assign weights globally.
  - Can be parallelized, in contrast to LSTM.
- E.g., the attention weights related to word “it_”:

[Slide from https://jalammar.github.io/illustrated-transformer/.]
Self-attention Mechanism

Input

Embedding

Queries

Keys

Values

Thinking

Machines

[Slide from https://jalammar.github.io/illustrated-transformer/]
Self-attention Mechanism

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 (\sqrt{d_k})

Softmax

Softmax

X

Value

Sum

Thinking

Machines

More…

More…

https://jalammar.github.io/illustrated-transformer/

[Slide from https://jalammar.github.io/illustrated-transformer/.]
Take home message

- Language models suffer from data sparsity
- Word2vec portrays language probability using distributed word embedding parameters
- ELMo, OpenAI GPT, BERT model language using deep neural networks
- Pre-trained language models or their parameters can be transferred to supervised learning problems in NLP
- Self-attention has the advantage over LSTM that it can be parallelized and consider interactions across the whole sentence
References

- Tensorflow. Vector Representations of Words: https://www.tensorflow.org/tutorials/representation/word2vec
- Jay Alammar. The Illustrated Transformer: https://jalammar.github.io/illustrated-transformer/