Neural Language Models

Marine Carpuat
Based on slides by Philipp Koehn (JHU)
Recap: Computation Graph

• To build a system, we only need to:
  • Define network structure
  • Define error/loss function
  • Provide data
  • (and set a few more hyperparameters to control training)

• Given network structure
  • Prediction is done by forward pass through graph (forward propagation)
  • Training is done by backward pass through graph (back propagation)
  • Based on simple matrix vector operations

• Forms the basis of neural network libraries
  • Tensorflow, Pytorch, mxnet, etc.
Language Modeling

- Goal: compute the probability of a sentence or sequence of words
  \[ P(E) = P(e_1,e_2,e_3,e_4,e_5...e_n) \]

- Related task: probability of an upcoming word
  \[ P(e_5|e_1,e_2,e_3,e_4) \]

- A model that computes either of these:
  \[ P(E) \text{ or } P(e_n|e_1,e_2...e_{n-1}) \]
  is called a **language model**.
Zipf’s Law
Zipf’s Law

- Even in a very large corpus, there will be a lot of infrequent words

- The same holds for many other levels of linguistic structure

- NLP/MT challenge: we need to be able to make predictions for things we have rarely or never seen
Toward a Neural Language Model
Representing Words

• “one hot vector”
  
  \[
  \text{dog} = [0, 0, 0, 0, 1, 0, 0, 0 \ldots] \\
  \text{cat} = [0, 0, 0, 0, 0, 0, 1, 0 \ldots] \\
  \text{eat} = [0, 1, 0, 0, 0, 0, 0, 0 \ldots]
  \]

• That’s a large vector! practical solutions:
  • limit to most frequent words (e.g., top 20000)
  • cluster words into classes
  • break up rare words into subword units
Language Modeling with Feedforward Neural Networks

Map each word into a lower-dimensional real-valued space using shared weight matrix C

Embedding layer

Bengio et al. 2003
An Output Layer to Predict Words

- Network will output a probability for each word in the vocabulary $V$

- Step 1: compute a score for each word in $V$
  $$ s = Wx + b $$

- Step 2: turn scores into probabilities using softmax function
  $$ p = \text{softmax}(s) $$

Where the probability of the $j$-th word in $V$ is
  $$ p_j = \frac{e^{s_j}}{\sum_j e^{s_j}} $$
Estimating Model Parameters

• Intuition: a model is good if it gives high probability to existing word sequences

• Training examples:
  • sequences of words in the language of interest

• Error/loss: negative log likelihood
  • At the corpus level error(\lambda) = − \sum_{E \text{ in corpus}} \log P_{\lambda}(E)
  • At the word level error(\lambda) = − \log P_{\lambda}(e_t | e_1 \ldots e_{t-1})
Language Modeling with Feedforward Neural Networks

Bengio et al. 2003
Word Embeddings: a useful by-product of neural LMs

- Words that occur in similar contexts tend to have similar embeddings
- Embeddings capture many usage regularities
- Useful features for many NLP tasks
Word Embeddings
Word Embeddings
Word Embeddings Capture Useful Regularities

Morpho-Syntactic
- Adjectives: base form vs. comparative
- Nouns: singular vs. plural
- Verbs: present tense vs. past tense
[Micholov et al. 2013]

Semantic
- Word similarity/relatedness
- Semantic relations
- But tends to fail at distinguishing
  - Synonyms vs. antonyms
  - Multiple senses of a word
Language Modeling with Feedforward Neural Networks

Bengio et al. 2003
Language Modeling with Recurrent Neural Networks

Figure by Philipp Koehn
Formalizing our Recurrent Language Model

\[ m_t = M_.c_{t-1} \]
\[ h_t = \begin{cases} \tanh(W_{mh} m_t + W_{hh} h_{t-1} + b_h) & t \geq 1, \\ 0 & \text{otherwise.} \end{cases} \]
\[ p_t = \text{softmax}(W_{hs} h_t + b_s). \]
Training

- Process 1\textsuperscript{st} example
- Update weights with backprop

- Process 2\textsuperscript{nd} example
- Update weights with backprop

- No feedback to previous history!
Training: Backpropagation Through Time

• Process a few examples

• Backpropagate through unfolded neural network
Practical Training Issues

- Compute parameter updates based on a “minibatch” of examples
  - instead of using one example at a time
- More efficient
  - matrix-matrix operations as opposed to multiple matrix-vector operations
- Can lead to better model parameters
  - middle ground between online and batch training

Figure by Graham Neubig
Practical Training Issues: vanishing/exploding gradients

\[
\frac{dl}{d_{h_1}} = \text{tiny} \quad \frac{dl}{d_{h_2}} = \text{small} \quad \frac{dl}{d_{h_3}} = \text{med.} \quad \frac{dl}{d_{h_4}} = \text{large}
\]

Figure 16: An example of the vanishing gradient problem.

Multiple ways to work around this problem:

- ReLU activations help
- Dedicated RNN architecture (Long Short Term Memory Networks)
Aside: Long Short Term Memory Networks

Figure by Christopher Olah
What do Recurrent Language Models Learn?

Cell sensitive to position in line:
The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy’s retreat and the soundness of the only possible line of action—the one Kutuzov and the general mass of the army demanded—namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all—carried on by vis inertiae—pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:
"You mean to imply that I have nothing to eat out of... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Figure from Karpathy 2015
What do Recurrent Language Models Learn?

Cell that turns on inside comments and quotes:
```c
// Duplicate LSM field information. The LSM rule is opaque, so
// not initialized.

static inline int audit_dupe_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    ret = 0;
    char *lsm_str;
    lsm_str = kstrdup(df->lsm_str, GFP_KERNEL);
    if (unlikely(lsm_str))
        ret = -ENOMEM;
    df->lsm_str = lsm_str;
    /* Our own (refcounted) copy of lsm rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("Empty rule for LSM \"%s\" is invalid!");
        ret = df->lsm_str;
    }
    return ret;
```

Cell that robustly activates inside if statements:
```c
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (ismember(current->notifier_mask, sig)) {
                if (!((current->notifier)(current->notifier_data))) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
            collect_signal(sig, pending, info);
        }
        return sig;
    }
```
What do Recurrent Language Models Learn?

• Can capture (some) long-distance dependencies

  After much economic progress over the years, the country has...

  The country, which has made much economic progress over the years, still has...
Deeper Models

- **Shallow**
- **Deep Stacked**
- **Deep Transition**
Neural Language Models

Summary

• A powerful tool for modeling language
  • Captures generalizations over words via embeddings
  • Captures some long-distance dependencies

• Build on computation graphs
  • some tricks required to train and predict efficiently

• Not just a building block of Neural MT systems
  • Have proved useful in statistical machine translation (Devlin et al. 2014)