Comprehensive Exam

Jeongmin Lee

May 4, 2018
Outline

1. Word Embedding for Text
2. Convolutional Neural Network for Text and Time-Series
3. Memory Network for Sequential Modeling and Question Answering
1. Word Embedding Methods

You shall know a word by the company it keeps. -J. R. Firth
In this part, we briefly look at the key ideas on word embedding methods and its variants on improvements as well as beyond modeling words
Represent Word as Vector

- Words can be represented as vectors
- The simplest form would be **one-hot** vector (bag of word)

```
<table>
<thead>
<tr>
<th></th>
<th>&quot;a&quot;</th>
<th>&quot;abbreviations&quot;</th>
<th>&quot;zoology&quot;</th>
<th>&quot;zoom&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Image Source: [https://ayearofai.com](https://ayearofai.com)

- Size of vector = **Cardinality of Corpus**
- There are an estimated 13 million words in English
  - With large corpus it will result in very sparse, high-dimensional vectors
- How do we represent it more efficiently?
Word2Vec (word-to-vector)

- **Key idea**: represent word as **dense real-valued** vector
  - learn two-layer neural network with the following objectives
  - use parameters of the network as the embedding for words

- **Two major approaches: Two Learning objectives**
  - Skipgram: given a word, predict its surrounding words
  - CBOW: given surrounding words, predict center word
Objective function (Skipgram)

- Maximize log probability of context words given the target word
  - $d = \text{the occurrences of all words in a corpus, } m = \text{size of context}$

$$J(\theta) = \frac{1}{d} \sum_{t=1}^{d} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t)$$

- $p(w_{t+j} | w_t)$ is the Softmax function of inner product of target and its context word
  - $u, v$ are input and output embedding matrices

$$p(w_{t+j} | w_t) = \frac{\exp(u_{t+j}^T \cdot v_t)}{\sum_{i=1}^{V} \exp(u_i^T \cdot v_t)}$$
Limitations of Embeddings

- Sensitive to superficial differences (dog/dogs)
- Unable to capture multiple meanings of word (financial bank / bank of a river)
- Not interpretable (negative values in embedding)
- Only able to embed words (not sentences, paragraphs)
Improving Embeddings

1. Insensitive to superficial differences
2. Capturing Multiple Meanings of Word
3. Add Interpretability
4. Embed Sentence, Paragraph, and Event
1. Insensitive to superficial differences

- **idea**: construct word embeddings from lower-level embeddings
- [Loung et al., 2013]
  - morpheme based embedding
    - morpheme: minimum meaning bearing unit
    - e.g., unfortunately = un + fortunate + ly
  - recursively construct embedding with RNN
- [Ling et al., 2015 a]
  - get embeddings of character first
  - then **dynamically generate** word embedding using LSTM
2. Capturing Multiple Meanings of Word

- **idea**: unrestrict number of embedding per word
- [Reisinger and Mooney, 2010]
  - a word's multiple context vectors are **clustered** to groups of similar ones
  - for each word, a set of meaning vectors are computed as a center of each cluster
- [Neelakantan et al., 2014]
  - extends [Reisinger and Mooney, 2010] to non-parametric version which does **not require the number of clusters** ahead
- [Athiwaratkun et al., 2017]
  - each word is modeled as a **mixture of the Gaussian** distribution
    - i.e., a word = multiple mean vectors and covariance matrices
    - maximize the similarity of distributions of nearby words in a sentence
3. Add Interpretability

- [Murphy et al., 2012]
  - factorize word-context count matrix with $L1(\text{sparsity reg.})$ and non-negativity constraints

- [Ling et al., 2015 b]
  - give different weights of context words when sum them in CBOW
  - weight is computed by additional feed-forward network (attention mechanism)
4. Embedding beyond Words

- [Lin et al., 2017]
  - learn **sentence-level** embedding
    - generate hidden states of each word in a sentence by LSTM
    - sentence embedding = **weighted sum** of hidden states of words (attention mechanism)
  - gains **interpretability** from weights on words for a task

  Example: weights visualized on 5 Star review text in Sentiment Classification task

- [Le and Mikolov, 2014]
  - learn embeddings for each **paragraph**
    - each paragraph has its own vector
    - predict target word given **paragraph AND context words** (CBOW)
• [Choi, et al. 2016]
  ○ learn embeddings of medical concepts (medication, diagnosis, procedure, etc)
  ○ make it to predict previous and next visit's concepts and given current visit's concepts (Skipgram)
Summary

1. Insensitive to superficial differences
   - Sub-word (Morpheme) Based Embedding
   - Character Level Embeddings
2. Capturing Multiple Meanings of Word
   - Multi-prototype (meaning) Embeddings
   - Embedding as Mixture of Gaussian
3. Add Interpretability
   - Non-negative Sparse Embedding
   - Weighting on Sum
4. Embed beyond Word
   - Embed Sentence
   - Embed Paragraph
   - Embed Discrete Events
2. Convolutional Neural Network for Text and Time Series Modeling
In this part, we first look into CNN's key operations **convolving** (sliding) and **pooling** and then we look at how CNN is applied to tasks in NLP and time-series modeling.
**Convolution**

- convolution is applying a **sliding filter (kernel)** to an input matrix (e.g., image) to extract feature
- with the filter, certain characteristic of a part of input can be extracted
- the filter is also part of parameter to be learned

![Convolution Example](http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution)

Pooling

- Pooling is extracting **smaller representative feature** among larger space
  - convolved feature has a property of **stationary**
    - characteristics of a part is **similar** to its surround parts
  - Pooling **summarizes** a group of convolved feature
- For pooling operation, **max** or **average** is used typically

Convolutional Neural Network (CNN)

- with non-linear activate functions (Tanh, ReLU), convolving and pooling operations compose CNN
- particularly, it has shown good performances in computer vision

Image source: http://www.wildml.com/
Why so powerful in computer vision? [Britz, 2015]

• Location Invariance
  ◦ convolution and pooling help to extract key information regardless of the key information's location
    ▪ invariance to rotation and scaling

• Compositionality
  ◦ each convolutional filter creates a local patch of lower-level features into a higher-level representation
  ◦ with "deep" network structure, important information for a task is composed bottom-up
    ▪ build edges from pixels, shapes from edges, and more complex objects from shapes
CNN for NLP and Time Series

- How CNN would be important in NLP
  - Example Problem: Sentiment Classification
- Text Classification
- Machine Translation
- Text Matching
- Discrete and Real-value Time series Prediction
Example Problem: Sentiment Classification

• Let's assume that we need to classify sentiment of sentences:

I hate this movie

I love this movie

very good
good
neutral
bad
very bad

very good
good
neutral
bad
very bad

Image source: [Neubig 2017]

• We might can do well with word embedding methods (CBOW or Skipgram)
Tricky Problem: Combinations of words

- What if we need to evaluate sentences like:

I don’t love this movie

There’s nothing I don’t love about this movie

- We need to model **combinations** of words

Image source: [Neubig 2017]
Simple Approach: Bag of N-Grams

- Word combinations will have its own embedding

\[
\text{sum}(I, \text{hate}, \text{this}, \text{movie}) = \text{probs}
\]

Image source: [Neubig 2017]

- **Problem**
  - No sharing of parameters between similar words/n-grams
  - need lots of those examples in train set
The idea of **convolution** comes to play at here

- Convolving with a filter (size=2) is same as soft 2-grams

- Parameters of each words will be **shared** and **combinations** can be modeled

Image source: [Neubig 2017]
CNN in NLP

- In 1990s and 2000s, CNN was not much used for NLP tasks
- [Waibel et al., 1990] provided idea of convolution
  - used for task of phoneme recognition from sound frequency data
- [Collobert and Weston 2011]
  - notable introduction of CNN in NLP
  - task: text-classification and Part-Of-Speech (POS) tagging
  - 1d-convolution after word-embedding layer
  - max-pooling over each embedding dimension
CNN for Text Classification (continued)

• [Kalchbrenner et al., 2014]
  ○ uses Dynamic K-Max Pooling to obtain top-K values in pooling layer
    ▪ K = a linear function of length of sentence and depth of a layer

• [Kim, 2014]
  ○ uses pretrained word embeddings on CNN
    ▪ pretrained embeddings were trained with large public data (Wikipedia)
CNN for Machine Translation

- Recent Neural Machine Translation (NMT) is based on **encoder-decoder** framework [Sutskever et al., 2014]
  - **Encoder** uses RNN to encode input sentence
  - **Decoder** uses another RNN to predict target sentence from encoded input sentence
- **Problem**: RNN cannot parallelizes procedure as it takes a sequence step-by-step

image: https://github.com/farizrahman4u/seq2seq
CNN for Machine Translation (continued)

Improve Throughput

• [Bradbury et al., 2017]
  ○ **combine** CNN and RNN to improve throughput (parallel processing)
    ▪ convolution over multiple time steps (words) all together
    ▪ **recurrent pooling** within LSTM component's forget and output gates (at gates, max pooling over multiple dimensions)

• [Gehring et al., 2017]
  ○ **CNN totally replaces** RNN in Encoder-Decoder model
  ○ issue: sense of sequence is lost
  ○ solution: **positional embedding**
    ▪ absolute position of word in a sentence = embedded vector
    ▪ sum of word and positional embedding are used
Use Character-Level Features

• [Yu and Koltun, 2016]
  ○ introduced concept of **dilation** in CNN
    ▪ gradually increases stride
    ▪ skip a fixed-number of inputs at regular interval
  ○ aggregate **multi-scale** contextual information

• [Kalchbrenner et al., 2016]
  ○ **character-level** processing of input and prediction
  ○ **stack** CNN encoder and CNN decoder
    ▪ predict single target character at a time by looking at encoder's dilated output and dilated CNN prediction so far
CNN for Text Matching

- Calculating distance of sentences
  - paraphrase identification
  - information retrieval (given query, rank document/sentence)
- [Bromley et al., 1993]
  - "Siamese Network" for signature identification (image matching)
  - input pair is passed to two same CNN and compute cosine similarity
- [Hu et al., 2014]
  - uses stacked two CNNs to compute distance between sentences
- [Yin et al., 2015]
  - multi-granular extension to [Hu et al., 2014]
  - compute distance of two sentences at each level's pooling layer
    - levels: unigram, short ngram, long ngram and sentence
  - concatenate each level's output together to generate final output
Time Series Prediction

- Predict Discrete Event
- Predict Real-Value Feature
Predict Discrete Event

• [Che et al., 2017]
  ○ from history of \textit{medication and diagnosis code} of patients
  ○ learn embeddings of the events using embedding layer followed by CNN
  ○ make it to \textit{predict risk} of diseases (multi-label)

• [Razavian et al., 2015]
  ○ input: multivariate time series on \textit{lab values} (real-value) of a patient
  ○ CNN with \textit{multiple resolution filter} (convolution) and \textit{max pooling} is used to extract features
  ○ trained to \textit{predict final diagnosis code} (multi-label)
Predict Real-value Feature

- [Ding et al., 2015]
  - CNN to extract important events from news document
  - follow up MLP is used to predict stock price of a company
- [Borovykh et al., 2017]
  - stacked dilation layers of convolution and max pooling
  - multivariate time-series prediction on financial data
Summary

• CNN
  ○ Convolving (Filter), Pooling (Summarize)
  ○ Location Invariance, Compositionality

• CNN for NLP
  ○ Text Classification
  ○ Machine Translation
    ■ Dilation
  ○ Text Matching

• CNN for Time Series Prediction
  ○ Discrete Event
  ○ Real-value Event
3. Memory Networks for Sequential Data

“What matters in life is not what happens to you but what you remember and how you remember it.”
— Gabriel García Márquez
In this part, I will cover **gist of Memory Networks** and then survey different types of Memory Networks for **tasks with sequential data**
Tasks with Sequential Data

- **Sequence Prediction**
  - given a sequence of entities, predict next one
  - e.g., Language Modeling

- **Sequence Memorization**
  - memorize randomly generated multivariate sequence (real-valued); Copy Task

- **Question-Answering (QA)**
  - facts or sentence is given in a sequence
  - when a question is presented, a model needs to give an answer
Example QA Task

John is in the playground.  
Bob is in the office.  
John picked up the football.  
Bob went to the kitchen.  
Where is the football? A: playground  
Where was Bob before the kitchen? A: office

Source: http://www.thespermwhale.com/jaseweston/icml2016/

• From the input sentences, it needs to find some important aspects related to the question
Example of Sequence Memorization Task

Source: [Graves et al., 2014]
Memory Networks

- Type of neural network that having external memory unit
- Motivated from limitations of RNN/LSTM
  - For modeling sequential data
    - hard to model longer-term dependencies
  - For Question-Answering (QA) task
    - Needs to memorize some parts of the input while reading input sentences
    - Accessing external knowledge bases is hard

Image credit: [Gulcehre and Chandar, 2017]
Input and Output Data

- Data type is **sequential discrete or real** variables
- **words** for **Language Modelings**
  - [Gulcehre et al., 2017], [Weston et al., 2014], [Sukhbaatar et al., 2015]
- **sentences** in **QA** task (multiple facts and a question)
  - [Gulcehre et al., 2016], [Graves et al., 2016], [Rae et al., 2016], [Weston et al., 2014], [Sukhbaatar et al., 2015], [Kumar et al., 2015], [Chandar et al., 2016], [Miller et al., 2016]
- **random number sequence** in **sequence copying/sorting tasks**
  - [Graves et al., 2014], [Gulcehre et al., 2016], [Graves et al., 2016], [Rae et al., 2016], [Gulcehre et al., 2017]
General Architecture

Image source: [Gulcehre and Chandar, 2017]
General Architecture

- Input module
  - embedding the input with word-embedding

- Controller (LSTM)
  - Reader: reads from memory and updates hidden states of controller $h_t$
  - Writer: updates contents of the memory

- Memory
  - Consists of multiple cells
  - each cell is a vector
    - $M \in \mathbb{R}^{p \times q}$
    - $p$: number of cells
    - $q$: size of vector cell
Reading Procedure

1. Generate a key for memory access

\[ k_t = f(x'_t, h_{t-1}) \in \mathbb{R}^{q \times 1} \]

2. Generate **weights** for memory cells based on **similarity with the key**

\[ w^r_t = \text{softmax}(M_t k_t) \in \mathbb{R}^{p \times 1} \]

3. Result of reading is a **weighted combination** of all the cells in the memory

\[ r_t = M_t^T w^r_t \in \mathbb{R}^{q \times 1} \]

\[ h_t = \text{LSTM}(x'_t, h_{t-1}, r_t) \]

- \( T \): matrix transpose
Writing Procedure

- Writing involves **erasing** old content and **adding** new content
  - Erase vector \( e_t = g(x', h_{t-1}) \in \mathbb{R}^{q \times 1} \)
  - write weights \( w_t^w \in \mathbb{R}^{p \times 1} \), from similar process of reading weights

1. First erase some content (\( i \): memory index, \( \mathbf{1} \): ones vector size \( q \))

\[
\tilde{M}_t(i) = M_{t-1}(i)\{\mathbf{1} - w_t^w(i)e_t\}
\]

2. Content to write:

\[
a_t = f'(h_t) \in \mathbb{R}^{q \times 1}
\]

3. Then add new content:

\[
M_t(i) = \tilde{M}_t(i) + w_t^w(i)a_t
\]
• This is architecture of **Neural Turing Machine** [Graves et al., 2014]
  ◦ memory is **flat** (not hierarchical)
  ◦ addressing mechanism uses **softmax**
    ▪ read/write operation is **not sparse**
    ▪ can be a challenge for larger memory
Variant approaches suggest to make Memory Network to be

- Hierarchical and Sparser read/write access
- Iterative Memory Access
- Separating Cells to Contents and Address
Sparser and Hierarchical Memory Access

- **Problem**: applying Softmax over whole memory cell is expensive

- [Rae et al., 2016]
  - access memory in hierarchical structure
  - get inner product of query and memory and select top-K cells
  - Softmax is applied to those top-K cells for read and write

- [Chandar et al., 2016]
  - extends to have clusters of memory cells
    - cluster memory cells based on distance of query and each cell
  - when accessing to memory is needed, select the closest centroid and performing Softmax within the cluster member cells
Iterative Memory Access

- **Idea**: before output final result, access memory multiple times
- [Kumar et al., 2015]
  - iteratively update current $h_t$ with multiple memory access
    - condition on the input and the result of previous access, generate final output
  - shown good performance at referring proper sentences on QA task:
    - e.g., Q: where football is? (A: room)
    - first iteration, find "john has a football" sentence
    - next iteration, find "john is in the room" sentence
Separating Cells to Contents and Address

- **Problem**: contents-as-address can be problematic
  - i. when external knowledge base is part of memory
  - ii. when size of each memory cell is growing larger
- [Miller et al., 2016]
  - separate memory cell into **address** (key) and **contents** (value)
  - contents of external knowledge base (WikiMovie) is in the memory
- [Gulcehre et al., 2016]
  - **learnable address vector** with **least recently used (LRU)** mechanism
    - **LRU**: when compute write weights, give more weights to least recently used cell locations
Summary

• Memory Network
• Hierarchical and Sparser read/write access
• Iterative Memory Access
• Separating Cells to Contents and Address
References
• References
• References

• References

○ [Choi et al., 2016] Choi, E., Bahadori, M. T., Searles, E., Coffey, C., Thompson, M., Bost, J., ... & Sun, J. (2016, August). Multi-layer representation learning for medical concepts. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1495-1504). ACM.


• References


• References


• References
• References


• References


• References
• References


• References
• References
