ELMO

Deep contextualized word representations

A R Shaarad, Prateek Sachan

Indian Institute of Science

April 26, 2019

Contents

Introduction

Embedding

- Computers understand only numbers.
- Need a way to represent every word.

Embedding

- Computers understand only numbers.
- Need a way to represent every word.
- Method 1:
  - Assign ID to each word in vocabulary
  - No relatedness between words

Embedding

- Computers understand only numbers.
- Need a way to represent every word.
- Method 1:
  - Assign ID to each word in vocabulary
  - No relatedness between words
- Method 2:
  - Sparse vector representation for every word
  - Word-context: count of words appearing in context window
  - Word-doc: count of words appearing in the document
  - Long, typically 20-50K for every word

Embedding

- Computers understand only numbers.
- Need a way to represent every word.
- Method 1:
  - Assign ID to each word in vocabulary
  - No relatedness between words
- Method 2:
  - Sparse vector representation for every word
  - Word-context: count of words appearing in context window
  - Word-doc: count of words appearing in the document
  - Long, typically 20-50K for every word
- Method 3:
  - Dense vector representation for every word
  - SVD based methods, Word2Vec, Glove
  - Short, typically 100-1000 for every word

Pre-trained Embedding

- Goal is to model
  - complex characteristics of word use (e.g., syntax and semantics)
  - how these uses vary across linguistic contexts

Pre-trained Embedding

- Goal is to model
  - complex characteristics of word use (e.g., syntax and semantics)
  - how these uses vary across linguistic contexts
- Word2Vec representation [Mik+13]
  - Unsupervised method
  - Increasing the similarity between words that appear in similar contexts
  - Performs well in semantic analogy tasks like synonyms, company-product relations, zip codes and cities, etc.
  - Use context only at time of training
  - Used as look-up tables at inference time, no context utilization.

Pre-trained Embedding

- Goal is to model
  - complex characteristics of word use (e.g., syntax and semantics)
  - how these uses vary across linguistic contexts
- Word2Vec representation [Mik+13]
  - Unsupervised method
  - Increasing the similarity between words that appear in similar contexts
  - Performs well in semantic analogy tasks like synonyms, company-product relations, zip codes and cities, etc.
  - Use context only at time of training
  - Used as look-up tables at inference time, no context utilization.
- Deep contextualized word representations
  - ELMo uses bi-Language model [Pet+18]
  - BERT uses bi-Transformer network [Dev+18]

Language model

- Probability distribution over a sequence of words
- Given a sequence of words $w_1, \ldots w_m$, the probability can be modelled as

$$P(w_1, \ldots w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots w_{i-1}).$$

Language model

- Probability distribution over a sequence of words
- Given a sequence of words $w_1, \dots w_m$, the probability can be modelled as

$$P(w_1, \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1, \dots w_{i-1}).$$

- Traditional methods
  - Count based
  - Estimate n-gram probabilities via counting and smoothing
  - Fail to estimate rare word probabilities
  - Finite history

Language model

- Probability distribution over a sequence of words
- Given a sequence of words $w_1, \dots w_m$, the probability can be modelled as

$$P(w_1, \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1, \dots w_{i-1}).$$

- Traditional methods
  - Count based
  - Estimate n-gram probabilities via counting and smoothing
  - Fail to estimate rare word probabilities
  - Finite history
- Neural language models
  - Use recurrent neural networks
  - Infinite history
  - Can handle rare words: Relatedness of embeddings

Model

Character Level CNN

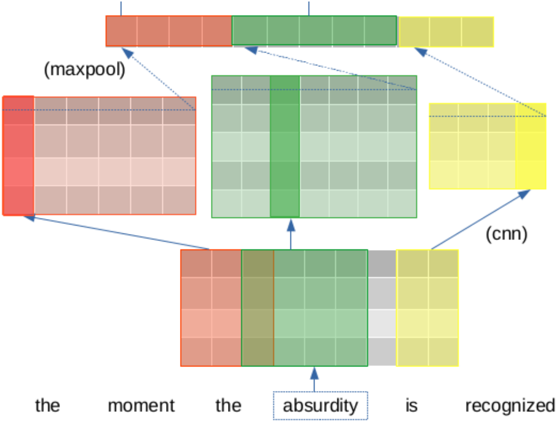

Figure: Char CNN [Kim+16]

Character Level CNN

- Character embedding dimension: 15
- Number of characters in a word: 32
- Filter size: 5
- Number of filters: 1000
- Pooling: Max pooling
- Nonlinearity: Tanh

Highway Network

- Outputs a combination of the input y and a transformed output
- The combination is itself determined by an affine transformation on the input
- $z = t \odot g(W_H y + b_H) + (1 - t) \odot y$
- $t = \sigma(W_T y + b_T)$: Transform gate
- $(1 - t)$: Carry gate

LSTM

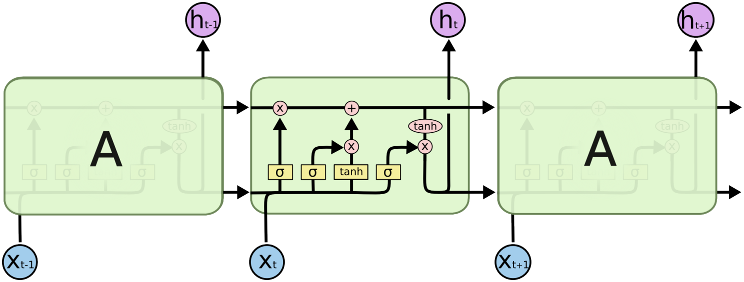

Figure: LSTM cell

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Bi Language model

- Predict next token given history

$$P(w_1, \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1, \dots w_{i-1}).$$

- Predict previous token given future context

$$P(w_1, \dots w_m) = \prod_{i=1}^{m} P(w_i | w_{i+1}, \dots w_N).$$

Bi Language model

- Predict next token given history

$$P(w_1, \ldots w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots w_{i-1}).$$

- Predict previous token given future context

$$P(w_1, \ldots w_m) = \prod_{i=1}^{m} P(w_i | w_{i+1}, \ldots w_N).$$

- In each LSTM layer, run two LSTMs, one in forward direction and one in backward direction
- To predict word w_k
  - Concatenate second layer forward LSTM output from word w_{k-1} and backward LSTM output of word w_{k+1}
  - Pass it through a dense layer
  - Apply softmax function to obtain probability distribution over words

Training

- Penn Treebank dataset
- $\sim$ 10000 unique words
- $\sim$ 1 million tokens
- 12 epochs
- Validation perplexity $\sim$ 89
- Batch size: 20
- Learning rate: 1

Embeddings

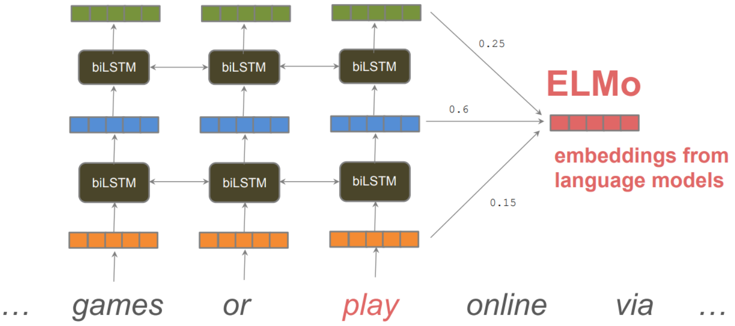

Figure: ELMO

Source: https://people.cs.umass.edu/ miyyer/cs585/lectures/06-neural-lms.pdf

Evaluation

Downstream task

- Sentiment analysis
- IMDB dataset
  - Classify movie reviews as positive or negative
- Model
  - Convolution layer with various filter sizes and max pooling
  - Fully connected layer
  - Regularization: Dropout
  - Loss function: Binary cross entropy
  - Adam Optimizer

Results

- Using ELMo Embedding
  - 79.31% accuracy
- Trained word2vec model using same Penn Treebank dataset
- Using word2vec Embedding
  - 83.44% accuracy

Conclusion

Conclusion

- Training a biLM to generate the word embedding requires large amount of data.
- This may be due to the fact that LM needs to see a large number of different sequences to generalize well.
- ELMo is a deep model and thus has a lot more parameters to train compared to word2vec model and thus require more data
- Smaller dataset is not sufficient to learn good language model for word embedding.

References

References I

- Tomas Mikolov et al. "Distributed Representations of Words and Phrases and Their Compositionality". In: **Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2**. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.
- Rafal Jozefowicz et al. **Exploring the limits of language modeling**. 2016. URL: https://arxiv.org/pdf/1602.02410.pdf.
- Yoon Kim et al. "Character-aware Neural Language Models". In: **Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence**. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2741–2749. URL: http://dl.acm.org/citation.cfm?id=3016100.3016285.
- Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: **arXiv preprint arXiv:1810.04805** (2018).
- Matthew E. Peters et al. "Deep contextualized word representations". In: **Proc. of NAACL**. 2018.