Natural Language Processing - Embeddings

How to Represent Text for Neural Networks

Ester Hlavnova (eh2757)

ECBM E6040 Neural Networks and Deep Learning, Spring 2019

Natural Language Processing with Deep Learning

Intro: Pre-Deep Learning NLP (No Word Embeddings)

I. **Static Embeddings** (MLP/RNN)
   A. Neural Probabilistic Language Model (NPLM)
   B. Word2Vec (*Word to Vector*)
   C. GloVe (*Global Vectors for Word Representation*)
   D. Character Embeddings

II. **Recurrent-based Contextualized Embeddings** (seq2seq with Attention)
   A. CoVe (*Contextualized Word Vectors)*
   B. ELMo (*Embeddings from Language Models*)

III. **Transformer-based Contextualized Embeddings** (Attention)
   A. BERT (*Bidirectional Encoder Representations from Transformers*)

Research Papers

Neural Probabilistic: **A Neural Probabilistic Language Model** by Yoshua Bengio (2003)

Word2Vec: **Distributed Representations of Words and Phrases and their Compositionality** by Tomas Mikolov, Ilya Sutskever, and Jeffrey Dean (2013)

GloVe: **Global Vectors for Word Representation** by Jeff Pennington, Richard Socher, Chris Manning (2014)

Character Embeddings: **Character-Aware Neural Language Models** by Yoon Kim, Yacine Jernite, David Sontag, and Sasha Rush (2015)

CoVe: **Learned in Translation: Contextualized Word Vectors** by Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher (2017)

ELMo: **Deep contextualized word representations** by Matthew Peters, Mark Neumann, Mohit Iyyer (2018)

BERT: **Pre-training of Deep Bidirectional Transformers for Language Understanding** by Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2018)

Intro: Pre-Deep Learning NLP (No Word Embeddings)

Many NLP tasks:

- Easy: Spell Checking, Keyword Search, Finding Synonyms,...
- Medium: Parsing Information from Websites or Documents,...
- Hard: Machine Translation, Semantic Analysis, Coreference, Question Answering,...

<u>How to represent words?</u> → A computer cannot understand strings, but does understand **vectors**

i. **Word Vectors**: one-hot encoding

ii. **Singular Value Decomposition** (SVD) methods:
   - Word-Document Matrix
   - Window based Co-occurrence Matrix

Intro: Pre-Deep Learning NLP (No Word Embeddings)

i. **One-hot encoding:**

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- V words in vocabulary → V-dimensional vector
- All words assumed independent: $\forall i \neq j, (w^i)^T w^j = 0$
- Can reduce the size of the space by reducing redundancy and sparsity (i.e. embeddings, making V entities represented within a space of size $<$ V)

ii. **SVD methods:**

- Word Document Matrix
  - Matrix V by M documents
  - Track words in document by 1
- Window based Co-occurrence Matrix
  - Matrix V by V
  - Counts co-occurrences in corpus

Applying SVD to X:

$$\begin{array}{c} \\ |V| \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix} \end{array} = \begin{array}{c} \\ |V| \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{c} \\ |V| \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{c} \\ |V| \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Reducing dimensionality by selecting first k singular vectors:

$$\begin{array}{c} \\ |V| \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} & & \\ & \hat{X} & \\ & & \end{bmatrix} \end{array} = \begin{array}{c} \\ |V| \end{array} \begin{array}{c} k \\ \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{c} \\ k \end{array} \begin{array}{c} k \\ \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{c} \\ k \end{array} \begin{array}{c} |V| \\ \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Intro: Pre-Deep Learning NLP (No Word Embeddings)

SVD methods have many problems:

- Quadratic cost to train due to SVD cost
- Matrix that SVD is performed on is high-dimensional, typically $10^6 \times 10^6$
- Requires the incorporation of some hacks on X to account for the drastic imbalance in word frequency

Some tricks can help to improve SVD methods:

- Reduce vocabulary with *stemming* (do=doing=does) and *stop-words* (delete repetitive words)
- Apply a *ramp window* – i.e. weight the co-occurrence count based on distance between the words in the document
- Use *Pearson correlation* and set counts with negative correlation to 0 instead of using raw counts

<u>But</u>: remains problematic and computationally expensive. → **This is Motivation for DL Embeddings!**

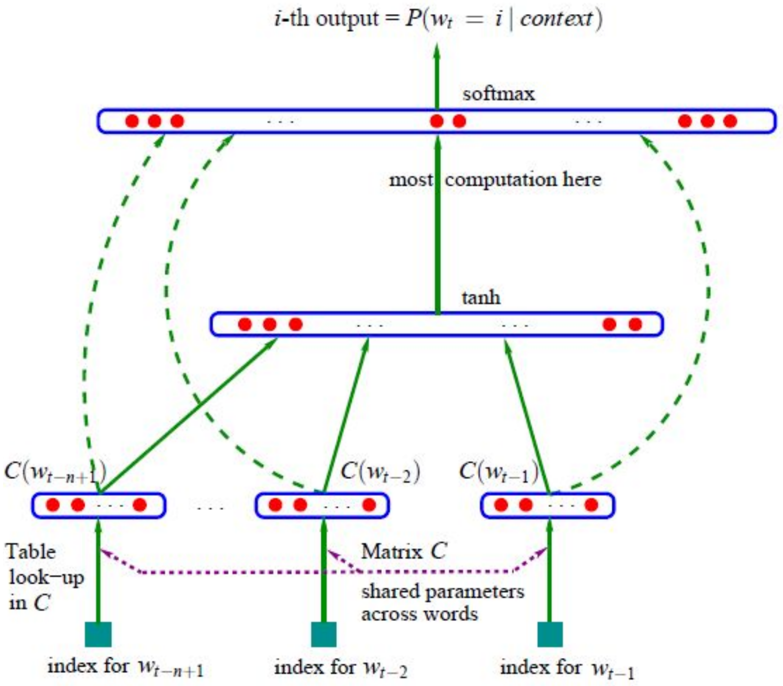
I - Static Embeddings (MLP/RNN)

A. Neural Probabilistic Language Model *(Bengio et al. - 2003)*

- Model probability distribution of next word given a sequence of N words:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t | w_{t-1}, \cdots, w_{t-n+1})$$

- Perform table lookup on the embedding matrix C, i.e. one-hot encoding x embedding matrix = embedding
- Bengio et al. suggest to replace MLP by LSTM to leverage the sequential nature of inputs
- Cost of computing softmax is proportional to the vocabulary size V



I - Static Embeddings (MLP/RNN)

B. Word2Vec *(Mikolov et al. - 2013)*

- Keep NPLM, but with *Continuous Bag-of-words* or *Skip-gram* model
- In both cases, avoid softmax bottleneck by *ranking* instead of *predicting*
- **Continuous Bag-of-words** (CBOW): from context predict a word

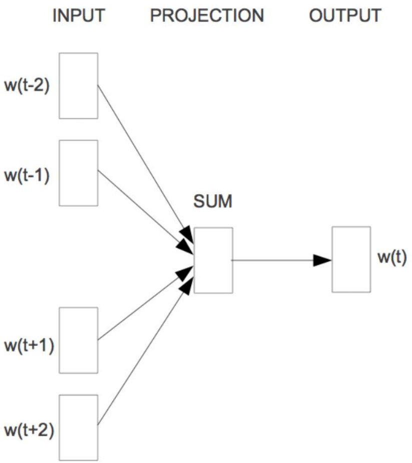
$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

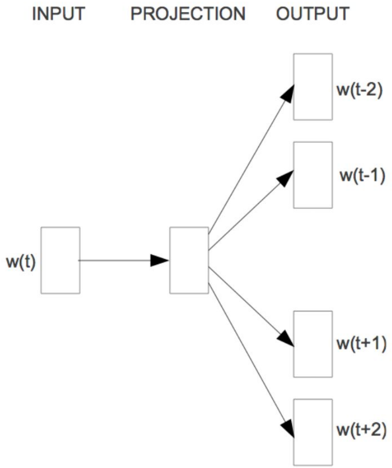
- **Skip-gram**: predict context from a word

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n\leq j\leq n, \neq 0} \log P(w_{t+j} \mid w_t)$$

- The main advantages of CBOW and Skip-gram:

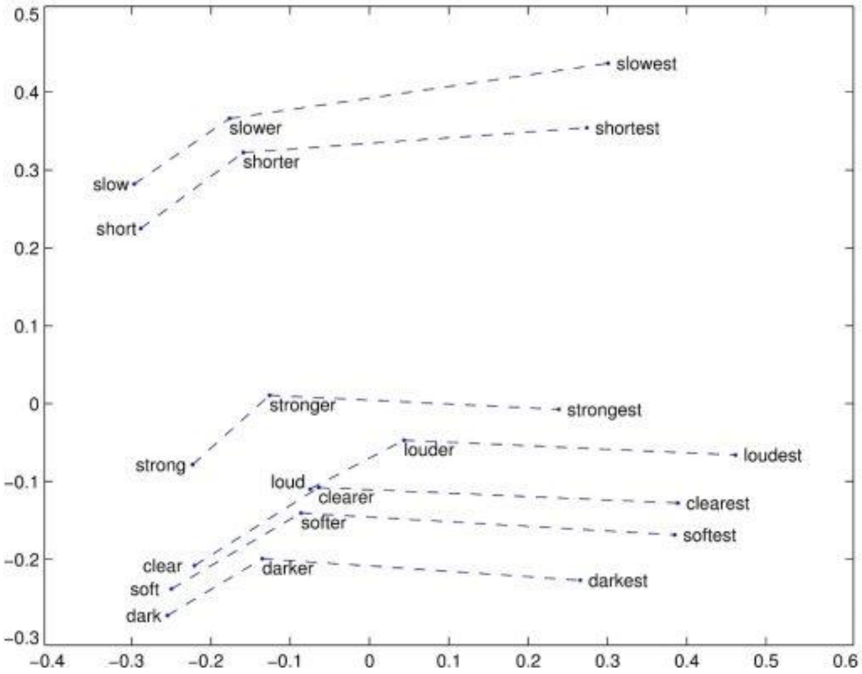
  i) no costly hidden layer (architecture) ii) additional context (task)





I - Static Embeddings (MLP/RNN)

C. **GloVe** *(Pennington et al. - 2014)*



- Ratio of co-occurrence probabilities of two words P_{ij} is main driver of language information, so they seek to predict P_{ij} to train embeddings

$$J = \sum_{i,j=1}^{V} f(X_{ij})\,(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log P_{ij})^2$$

- w_i represents the embedding of word i and f, a weighting function that assigns relatively lower weight to rare and frequent co-occurrences
- Works on word context co-occurrences
- Learns interesting words algebra like the famous: king − man + woman = queen

I - Static Embeddings (MLP/RNN)

D. Character Embeddings *(Kim et al. - 2015)*

- Instead of embedding words, why not try with characters?
- Generally embedding of characters ngrams
- 2 fundamental advantages:
  - Reduce drastically the vocabulary size
  - Can deal with out-of-vocabulary unlike word embeddings
- Can also be concatenated with word embeddings

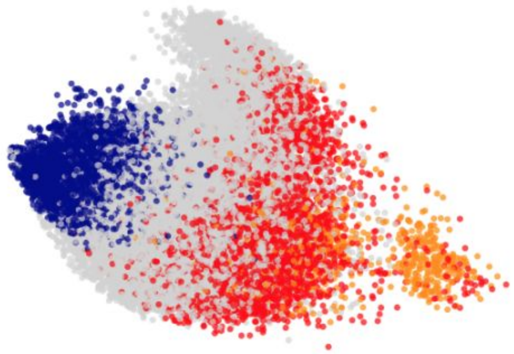


Figure 2: Plot of character n-gram representations via PCA for English. Colors correspond to: prefixes (red), suffixes (blue), hyphenated (orange), and all others (grey). Prefixes refer to character n-grams which start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.

II - Recurrent Contextualized Embeddings (seq2seq Attention)

A. **CoVe** *(McCann et al. - 2017)*

- Start from simple statement that word2vec and GloVe are static → i.e. the contextualized information has to be learnt by the network
- Train an encoder/decoder architecture with bidirectional LSTM and attention where encoder is used to encode word vectors
- Train on Machine Translation task and aim to leverage transfer learning
- Inputs are concatenation of GloVe and Char embeddings from Salesforce
- Uses top layers LSTM hidden-states
- Outperforms GloVe on various downstream NLP tasks, i.e. sentiment analysis, name entity recognition

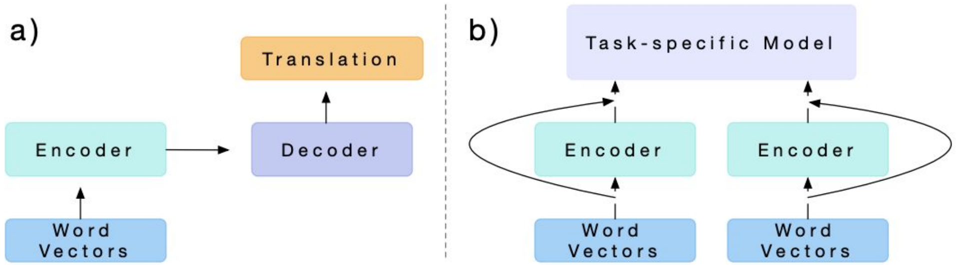


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

II - Recurrent Contextualized Embeddings (seq2seq Attention)

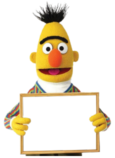
B. **ELMo** *(Peters et al. - 2018)*

- Train deep embeddings by tranning 2 biLSTM language models and by using *all* LSTM encoder layers
- Advantage over CoVe - can be trained on large amount of data
- Char CNN to build initial word representations instead of mix of pretrained embeddings
- Obtain SOTA results on 6 tasks where performance gain comes only from ELMo:
  - Question Answering
  - Textual entailment
  - Semantic role labeling
  - Coreference resolution
  - Named entity extraction
  - Sentiment analysis

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

III - Transformer Contextualized Embeddings (Attention)

A. **BERT** *(Devlin et al. - 2018)*



- Replaces seq2seq LSTM attention model by a transformer (i.e. recurrence by full attention)
- Bidirectionality works differently: no concatenation of independent unidirectional representations
- *Masked* language model and advanced location Embeddings (used by Transformers)

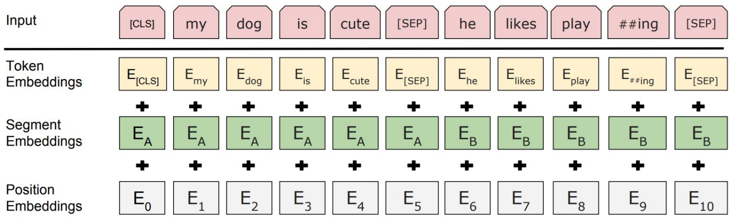
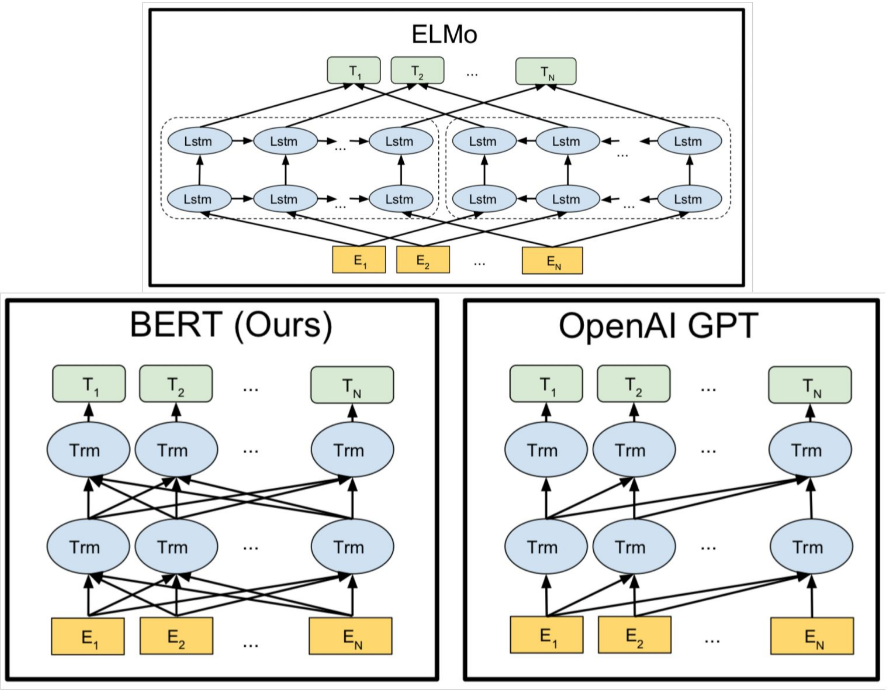


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.



References

- Bengio, Y., Ducharme, R. & Vincent, P. A neural probabilistic language model. In Proc. Advances in Neural Information Processing Systems 13 932–938 (2001).
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed representations of words and phrases and their compositionality. In Proc. Advances in Neural Information Processing Systems 26 3111–3119 (2013).
- R. Jeffrey Pennington and C. Manning. Glove: Global vectors for word representation. 2014.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. CoRR, abs/1508.06615.
- Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In NIPS 2017.
- M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. NAACL, 2018.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Main Resources

- Stanford CS224d Deep Learning for NLP notes
- Blog on Word2Vec and GloVe
- Contextualized Word Vectors from CS224d