CS224n: NLP with Deep Learning **Winter 2019**

Lecture 2

Course Coordinator: Prof. Chris Manning *Scribes: Akash Gupta*

From last lecture -

- Build analogies by doing simple algebraic operations on word vectors. For e.g. - $v(king) - v(man) + v(woman) = v(queen)$
- Can Project in 2D space using PCA to see which word vectors are close-by but beware that reducing the dimensions might also change some of these relationships. So, some similar words might not be close due to the fact that their relationship was determined by some other principal components.

Word2Vec - Real Scenario for predictions: We have a matrix referred to as U whose each row is a word vector of outside words. And we have another matrix V whose each row contains a center word. So, we do the below calculation output a probability distribution for each center word:

$$U = \begin{bmatrix} *****\\ *****\\ *****\\ *****\\ ***** \end{bmatrix}, V = \begin{bmatrix} *****\\ *****\\ *****\\ *****\\ ***** \end{bmatrix}, Uv_4 = \begin{bmatrix} *\\ *\\ *\\ *\\ * \end{bmatrix}, Softmax(Uv_4) = \begin{bmatrix} *\\ *\\ *\\ *\\ * \end{bmatrix}$$

NOTE:- Same predictions will be given to every position. So, the idea here is not to make accurate predictions but we "just" want the model to assign reasonably higher probabilities to all the words that occur in the context.

Observation:- You might wonder that words like the, and, a, ... occur too often and their probabilities will be higher as compared to rest. The ans is "Yes". All word vectors have a very strong word probability component that reflects that. A paper by Sanjeev Arora at Princeton discusses how to deal with high frequency by removing the first biggest component which is the high frequency vector and removing that would help to make semantic similarities.

Gradient Descent - Use Batch Gradient Descent - for faster optimization on GPUs and less noise compare Stochastic G.D.

Gradient Descent for word vectors - Iteratively take gradients for each window using SGD but this would result in sparse parameter update because each window has only $2m + 1$ words and so $\nabla_\theta J_t(\theta)$ is sparse. Solution: Either do sparse matrix updates to only update rows that actually appear or keep a hash for word vectors.

Word2Vec - More details -

- Why 2 vectors (U and V)? - Easier optimization by avoiding square terms. Can take average at the end.
- Two model variants:
  - Skip-gram - Predict the outside ("context") words (position independent) given the center word
  - Continuous Bag of Words (CBOW) - Predict center word from (bag of) context words.

- Additional efficiency in training - Negative Sampling - Idea is to train a binary logistic regression for a true pair (center word and word in it's context window) versus several noise pairs (the center word paired with a random word)

Skip-gram model with negative sampling:

$$J_{neg-sample}(o, v_c, U) \ = \ -log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} log((\sigma(-u_k^T v_c)))$$

- We take k negative samples (using word probabilities)
- Maximize the probability that real outside words appear and minimize probability that random words appear around center word.
- $P(w) = U(w)^{3/4}/Z$: U(w) is the Unigram distribution, Z is the normalization constant. This provides less frequent words being sampled more often

Another approach: co-occurrence matrix - (Window- based or full document)

- Window based - common window size (5 -10) and symmetric (irrelevant if occurs either left or right)
- Problems with simple co-occurrence vectors.
  - Increase in size of vocabulary.
  - Very high dimensional - Lot of storage.
  - Models less robust.

Dimensionality Reduction on X: Use Singular Value Decomposition which factorizes X into $U \sum V^T$, where U and V are orthonormal and $\sum$ is diagonal containing sigular values with a decreasing trend.

$->$ Also seen that transforming counts gave a very good estimate of the word vectors.

$->$ Two pieces of work - Count-based vs. direct prediction.

$->$ **Encoding meaning in vector differences** by Pennington, Manning - Ratios of co-occurrence probabilities can encode meaning components. How to do that?
A) Log-bilinear model. - Dot products equal to log of co-occurrence prob which makes difference of the word vectors equal to ratio of co-occurrence prob - GloVe model

Word Senses - Words have lots of meanings. Develop a model that takes all the meanings into account - Sanjeev Arora paper.