

C224N: Stanford

Chris Manning:

## Lecture 1: Introduction and Word Vectors

- An understanding of the effective modern methods for deep learning
- Basics first, then key methods used in NLP:
  - Recurrent networks, attention, etc.
- A big picture understanding of human language and the difficulties in understanding and producing them
- An understanding of an ability to build systems for some of the major problems in NLP
  - word meaning, dependency parsing, machine translation, question answering
- Language isn't a formal system. It's a glorious chaos
- Humans are better because of the language and human communication network approach
- Manning vs LeCun (Humans are better)
  - Writing takes knowledge spatially throughout the word and temporally through time

1) How do you represent the meaning of a word?

Brigham and  
meaning

Definition: meaning

→ idea that is represented by a word, phrase etc.

→ idea that a person wants to express by using words, signs, etc.

→ idea that is expressed in a work of writing, art etc.

Linguistic way of thinking of meaning:

Signifier (symbol)  $\leftrightarrow$  signified (idea of thing)

= denotational semantics

Common solutions to have usable meaning

Use Wordnet (nltk), a thesaurus containing lists of synonym sets and hypernyms

But that's not ~~terribly good at anything~~

problem: can't add pt

will specialize and refine

## Representing words as discrete symbols

Traditional NLP (NLP till 2012) regards words as discrete symbols → localist rep.  
were represented as one-hot vectors  
⇒ but words are infinite.  
one possible solution to find the synonyms  
can be word similarity table

Distributional semantics: A word's meaning is given by the words that frequently appear close-by  
⇒ when a word  $w$  appears in a text, its context is the set of words that appear nearby (within a fixed-size window)  
Word Vectors: We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

Word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

Word meaning as a neural word vector:

(Yoshua Bengio's work)

Word2Vec: (Mikolov et. al. 2013)

### Idea:

- i) We have a large corpus of text
- ii) Every word in a fixed vocabulary is represented by a vector
- iii) Go through each position  $t$  in the text, which has a center word  $c$  and context ("outside") words  $o$
- iv) Use the similarity of the word vectors for  $c$  and  $o$  to calculate the probability of  $o$  given  $c$  (or vice versa)  
$$P(w_{+j}/w_t)$$
- v) Keep adjusting the word vectors to maximise this probability

objection function (cost or loss)

for each position  $t = 1 \dots T$  predict context words within a window of fixed size  $m$ , give center word  $w_t$

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

slide  $\rightarrow$  all variables to be optimized

The objective function  $J(\theta)$  is the (average) negative

log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = \theta \cdot \underbrace{\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}}}_{\substack{\text{minimizing} \\ \text{average}}} \log P(w_{t+j} | w_t; \theta)$$

Minimizing objective  $\Leftrightarrow$  Maximizing predictive accuracy

How to calculate  $P(w_{t+1} | w_t, \theta)$

Answers: Two vector representations

•  $v_w$  when  $w$  is center word

•  $u_w$  when  $w$  is a context word

center  $c$   
context  $w$

$$P(\theta | c) = \frac{\exp(u_w^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} \rightarrow \begin{array}{l} \text{compare similarity of } \\ \theta \text{ and } c \\ \text{normalize over entire vocab} \end{array}$$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} = p_i$$

The softmax function maps arbitrary values  $x_i$  to a probability distribution  $p_i$

$\Rightarrow$  amplifier probability of largest  $x_i$  while still assigning some to smaller  $x$

Training a model by optimizing parameters

$$\frac{\delta}{\delta V_c} \log \frac{\exp(u_0^T v_c)}{\sum_{w=1}^m \exp(u_w^T v_c)}$$

$$= \frac{\delta}{\delta V_c} \log \exp(u_0^T v_c) - \frac{\delta}{\delta V_c} \log \sum_{w=1}^m \exp(u_w^T v_c)$$

$$= \frac{\delta}{\delta V_c} u_0^T v_c +$$

negative word gensim  
visualization. There's some ambiguity with names.

$$= u_0^T v_c - \frac{1}{m} \sum_{w=1}^m u_w^T v_c$$

$$\frac{d}{dv_c} \log \sum_{w=1}^V \exp(u_w^T v_c)$$

$$= \frac{1}{\sum_{w=1}^V \exp(u_w^T v_c)} \cdot \sum_{x=1}^V \frac{d}{dv_c} \exp(u_x^T v_c)$$

$$\sum_{x=1}^V \exp(u_x^T v_c) \frac{d}{dv_c} u_x^T v_c$$

$$\sum_{x=1}^V \exp(u_x^T v_c) u_x$$

$$\frac{\delta}{\delta v_c} \log P(O|C) = u_o - \frac{\sum_{x=1}^V \exp(u_x^T v_c) u_x}{\sum_{w=1}^V \exp(u_w^T v_c)}$$

$$= u_o - \sum_{x=1}^V \frac{\exp(u_x^T v_c)}{\sum_{w=1}^V \exp(u_w^T v_c)} u_x$$

actual context word

$u \rightarrow$  outside  
 $v \rightarrow$  center

$v, v^T$   
softmax( $v, v^T$ )  
probs

$$= u_o - \sum_{x=1}^V P(x|C) u_x$$

1 observed representation of each word of context word

expected context word

probability of it in current model

gem sim (word similarity packet)

LDA

negative similarity

## Lecture 2 (Word Vectors & Word Senses) (4/10/2020) (2)

The word vectors are represented as rows

We want a model that gives a reasonably high probability estimate to all words that occur in the context (that's why we are using softmax)

Optimization: Gradient Descent

minimize  $J(\theta)$

$$\theta^{\text{new}} = \theta^{\text{old}} - \alpha \nabla J(\theta)$$

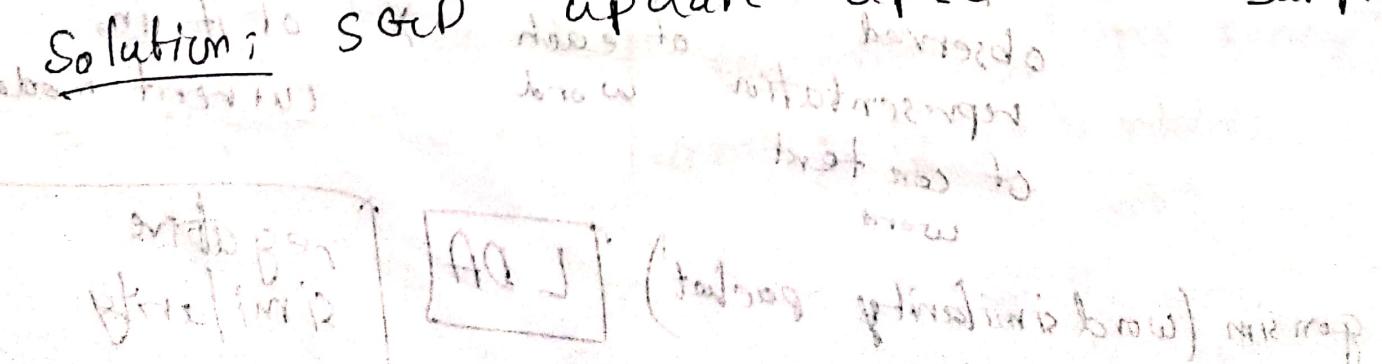
Stochastic Gradient Descent

Since  $J(\theta)$  is very expensive to compute here,

→ Very bad idea for pretty much all uses

SGD update after each sample

Solution:



## Stochastic gradient with wordvecs

We might only update the word vectors that actually appear.

Solution: Either we need sparse matrix update operations to only update certain rows of full embeddings matrices  $U$  and  $V$ , or we need to keep a hash of word vectors if you have millions of words and do distributed computing. It is important to not have to send gigantic updates around.

### Why two vectors?

Easier optimization. Average both at the end

Two model variants:

1) Skip-Gram ( $SR$ )  $\Rightarrow$  predict context words given center word

2) Continuous Bag of Words ( $(CBOW)$ )  $\Rightarrow$  predict context words from (bag of) context words

Naive Softmax

Skip-gram with Neg-sampling: Normalization factor is too computationally expensive

Idea of Negative sampling:

Train binary logistic regression for a P

true pair (center word and word in its context)

versus several noise pairs (center word paired with a random word) → then try to give them as lower prob as possible

Paper: "Distributed Representations of Words and

Phrases and their Compositionalities" (Mikolov 2013)

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J_t(\theta)$$

$$J_t(\theta) = \log \delta(u_b^T v_c) + \sum_{j=1}^k p(w_j) \left[ \log \delta(-u_j^T v_c) \right]$$

→ We maximize the probability of two words

co-occurring in the first log

$$J_{\text{neg-sample}}(\theta, v_c, u) = -\log (\delta(u_b^T v_c)) - \sum_{k=1}^K \log (\delta(-u_k^T v_c))$$

We take k negative samples (using word probs)

- Maximize probability that word appears outside random words appear around center word

$$P(w) = \frac{W(w)}{\text{Unigram dist}}$$

count of word / total words

makes less frequent word to be sampled more often

But why not capture co-occurrence counts directly?

With co-occurrence matrix  $X$

- 2 options: windows vs. full doc
- Windows: similar to word2vec, use window around each word → captures both syntactic (POS) and semantic information
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to "Latent Semantic Analysis"

- Example: window base co-occurrence matrix
- Window length 1 (more common - 10)
  - Symmetric (irrelevant whether left or right context)

### Problems with simple co-occurrence vecs

- Increase the size with vocab
- very high dimensional: requires a lot of storage
- subsequent classification model have sparsity issues
- models are less robust at scaling up.

Solution: Low dimensional vectors

store "most" of the important info in a fixed small number of dimensions; a dense vector usually 25-100 dimensions, similar to word2vec

Reducing dimensionality

## Method 1: Dimensionality Reduction of $X$ (HW1)

Singular Value Decomposition of co-occurrence matrix  $X$ .

Factorizes  $X$  to  $U\Sigma V^T$  where  $U$  and  $V$  are orthogonal

If we take  $k$  values then, we ignore the least frequent context vectors

$\hat{X}$  is the best rank approx to  $X$ , in terms of least squares

Problems: Expensive to compute for large matrices.

[Play around with it in homework]

Hacks to  $X$  (Rohde 2005)

Scaling the counts in the cells help a lot.

- Problems function word (the, he, has) are too frequent  $\rightarrow$  syntax has too much impact.  
→ Some fixes  $\min(X, t) + \approx 100$   
→ ignore all

- Ramped windows that count closer words more
- Use pearson correlation instead of counts, then set negatives to 0

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}}$$

### Count-based vs direct prediction

- LSA, HAL (Lund & Burgess)
- CoAE, Hellinger-PCA (Rohde) (Lebret & Collobert)

- Skip-gram/CBOW (Mikolov)
- NNLM (~~RNN~~) HLBL (Collobert)  
RNN (Bengio, Huang, Mnih)

| Pros | <ul style="list-style-type: none"> <li>Fast training</li> <li>Efficient usages of statistics</li> </ul>  |
|------|--|
| Cons | <ul style="list-style-type: none"> <li>Primarily used to capture word similarity</li> <li>Disproportionate importance given to large counts</li> </ul> |

| Cons | <ul style="list-style-type: none"> <li>Scales with corpus size</li> <li>Inefficient usage of stats</li> </ul>                                |
|------|--|
| Pros | <ul style="list-style-type: none"> <li>Generate improved performance</li> <li>Can capture complex patterns beyond word similarity</li> </ul> |

Encoding meaning in vector differences [EMNLP 2014]

(Pennington)

[Combining the best of both worlds]

Crucial insight: ratios of co-occurrence probability  
can encode meaning components

Capturing ratios of co-occurrence probs as  
linear meaning

$$w_i w_j = \log P(i/j)$$

$$w_x (w_a - w_b) = \log \frac{P(x/a)}{P(x/b)}$$

$$J = \sum_{ij=1}^V f(x_{ij}) (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log x_{ij})^2 \quad [\text{GloVe}]$$

- fast training

- Scalable huge corpora

- Good performance even with small corpus and  
small vectors

## Evaluate Word Vectors

### General Evaluation in NLP: Intrinsic vs Extrinsic

#### Intrinsic:

- Evaluation of a specific/intermediate subtask
- Fast to compute
- Helps to understand that system
- Not clear if really helpful unless correlation to real task is established.

#### Extrinsic:

- Evaluation on a real task
- Can take long time to compute accuracy
- Unclear if the subsystem is the problem or its interaction ~~is established or~~ <sup>other</sup> subsystems
- If replacing exactly one subsystem with another improves accuracy → winning

## Intrinsic word vector evaluation

- Word vector analogies

$a:b :: c:?$

man: woman :: king: ?

$$d = \operatorname{argmax}_i \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\|}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactical analogy questions.
- Discarding the input words from the search!
- Problem: what if the info is there but not linear.

## for Glove

- Good dimension is  $\sim 300$
- Asymmetric context rate not as good
- This might be different for downstream tasks
- Window size of 8 around each center word is good for glove vectors

## Recent works on dimensionality of Word Embedding [NeurIPS 2018]

- Wikipedia is better than news text

- Word vector distances and their correlation with human judgment.

Example Wordsim353 (check out)

Cosine distance for similarity judging

Word senses and word sense ambiguity

→ Most words have lots of meanings

Does one vector capture all these meanings?

Improving Word Rep via Global Context (Huang et al 2012)

Ideal Cluster ~~around~~ windows around words, word retain with each word assigned to multiple different clusters bank<sub>1</sub>, bank<sub>2</sub>

Linear Algebraic Structure of Word Senses with Applications to Polysemy (Arora, TACL 2018)

Different senses of a word reside in a linear superposition (weighted sum). In standard word embedding a like wordvec

$$V_{\text{pike}} = d_1 V_{\text{pike}_1} + d_2 V_{\text{pike}_2} + d_3 V_{\text{pike}_3}$$

$$d_1 = \frac{f_1}{f_1 + f_2 + f_3} \quad f = \text{frequency}$$

- Because of ideas from sparse coding we can actually separate out the senses.

Extrinsic Word Vector Evaluation:

All subsequent tasks in this class

→ Good Word vectors would help in named entity recognition: