

Word Embeddings

Jelke Bloem

Text Mining
Amsterdam University College

March 11, 2025

Assignments

- Assignment 2 (word embeddings): 25/03
- Reading assignment Word2Vec: 14/03

Questions/discussion reading assignment 1

- What are different ways we can allocate vector dimensions for different purposes, besides the ones listed in the reading (e.g a dimension being a class like days of the week with binary values)?
- Smith highlights a disadvantage of dimensionality reduction is that individual dimensions lose their specific, interpretable meanings. The word's meaning is then distributed across the entire vector rather than being concentrated in specific interpretable features, leading to the term “distributed representations.” Are distributed representations therefore a negative consequence of the dimensionality reduction of word vectors?
- How can a LLM like ChatGPT train so fast over such a short period of time?(since the learning curve is exponential)

Questions/discussion reading assignment 1

- If a statistical method for natural language processing can achieve extremely similar results to a human, albeit using different methods, should we view this as the same understanding of language?
- What (else) is required of a language model/technology to ever achieve a full understanding of human language?
- Given that LLM's, although able to predict/recreate language very well, will always lack the element of 'the human experience' and therefore will never (?) be able to 100% accurately predict language outcome, at what scale do these inconsistencies become noticeable? Like for example, given an emotionally charged question, what percentage of the time is the human psyche too unpredictable to be predicted.

Overview

1 Recap on Word Vectors

2 Word2Vec

3 Word2Vec Expanded

Recap on Word Vectors

Vector Semantics

“The meaning of a word is its use in the language.” Wittgenstein, 1953.

- ① Vector semantics combines two intuitions:
 - ▶ **Distributional approach**: define a word by the contexts it occurs into.
 - ▶ **Vectorization**: use vectors to represent word meaning.
- ② **Feature engineering** for NLP: word vectors are used as features for other tasks.
- ③ (Word) vectors are usually referred as (word) **embeddings** in modern neural network literature.

Co-occurrences

...ound and sonic power of a [new electric guitar played through] a guitar amp has play...
...[Some electric guitar models feature] piezoelectric pickups...
...[Playing guitar with a] pick produces a bright sound ...
...ings, he is known for [playing fretless guitar in his] performances...
...the neck of [a classical guitar is too] wide and the normal position ...
...t in the centre of Bristol [playing the piano , I was] punched in the head while, a...
...r in Houston, Texanstagram [playing the piano in his] flooded home after Hurrican H...
... some supplies, he stopped to [play the piano that was] sitting in knee-high water ...
...te and one black, who [played classical pianos together]...
...The [first electric from the] late 1920s used metal strin...
...technologies, for example [the electric car and the] integration of mobile commun...
...study had each driver of [each electric car drive unimpeded], perform a task whil...
...Honda to commence testing of [their new car and the] American was no doubt more t...
...many design considerations for [the new car were "safety] innovations, performanc...
...would be possible if almost [all private cars requiring drivers], which are not in ...
... who donate to groups [providing private school scholarships have] written pieces att...
... that students participating [in private school choice programs] graduate high school...
...s in the establishment of this [new high school , named the] Gavirate Business School...
...Anna heads into her [final high school year before] university wanting somet...
... but he can prevent them from [playing at school]

Word-Context matrix

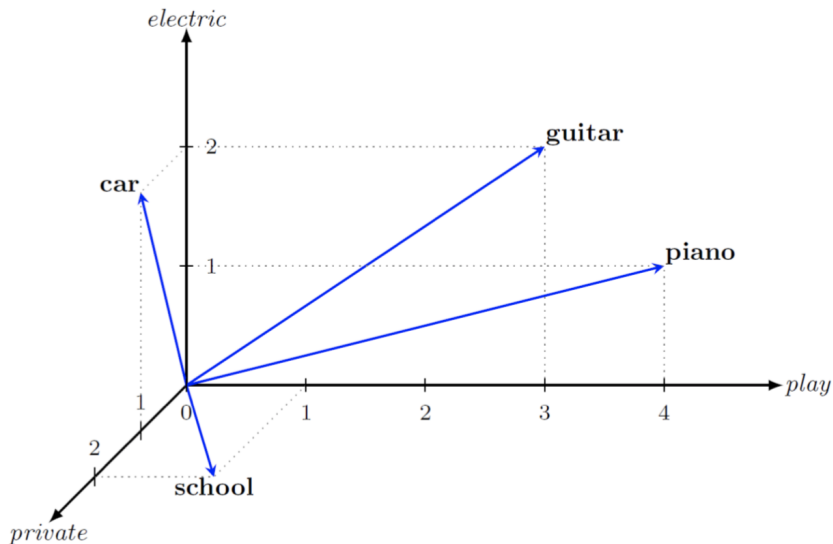
- We have a set of words V and a set of contexts they occur into C , taken from our corpus of documents. X in this case is a $|V| \times |C|$ matrix with word occurrences in contexts.
- The most intuitive context are co-occurrences with other words in V , within a certain **window**. In this case, X would be a $|V| \times |V|$ matrix.

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Figure 6.5 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Credit: J&M, ch. 6.

Vectors



Families of vectors

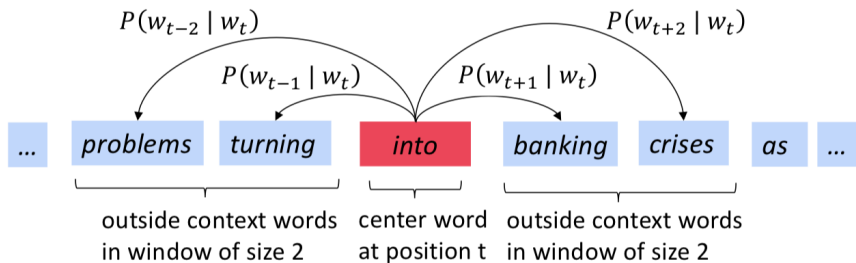
- **Sparse vectors:** many zero values and high-dimensional spaces. E.g., weighted co-occurrence matrices.
- **Dense vectors:** no zero values and comparatively smaller-dimensional spaces.
 - ▶ Dimensionality reduction (Singular Value Decomposition, Random indexing, Non-negative matrix factorization).
 - ▶ **Neural-network inspired** (Word2Vec, GloVe, BERT and many more): we start today.
 - ★ Static word embeddings (Word2Vec, GloVe, FastText)
 - ★ Contextual word embeddings (BERT, ModernBERT, XLM-RoBERTa, XLNet, GPT-2 (unidirectional))

Word2Vec

Intuition

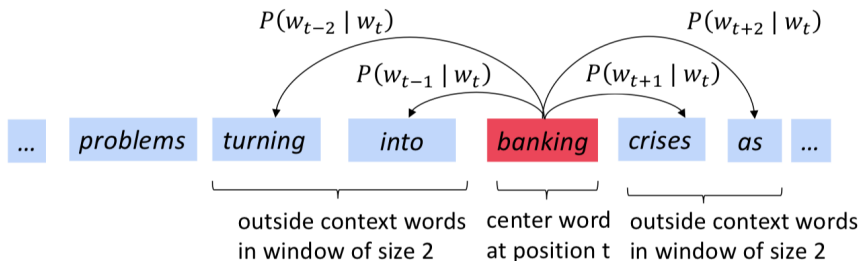
- **Word2Vec**: a framework for learning dense word vectors.
- Idea:
 - ① We have a large corpus of text.
 - ② We want each word in the vocabulary to be represented by a vector.
 - ③ We can go through the corpus and establish a *context* o for every *center/focus word* c , using a certain window/span.
 - ④ **We use the similarity of the word vectors c and o to calculate the probability of context words o given c .**
 - ⑤ **We keep adjusting word vectors until our predictions are good.**

Words in context



Credit: Stanford CS224N.

Words in context



Credit: Stanford CS224N.

Words in context as data

Source Text

Training Samples

<div>The quick brown fox jumps over the lazy dog.</div>	→	(the, quick) (the, brown)
<div>The quick brown fox jumps over the lazy dog.</div>	→	(quick, the) (quick, brown) (quick, fox)
<div>The quick brown fox jumps over the lazy dog.</div>	→	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
<div>The quick brown fox jumps over the lazy dog.</div>	→	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Credit: [http:](http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model)

[//mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model](http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model).

The model

- Our task, for every c (center), o (context) pair, is to estimate high probabilities for:

$$p(w_o|w_c)$$

- *The model parameters are the word embeddings w .*
- For each word position $t = 1 \dots T$, we predict context words within a windows of size m , given the center word w_t (at each position):

$$L(\mathbf{w}) = \prod_{t=1}^T \prod_{-m \leq j \leq m; j \neq 0} p(\mathbf{w}_{t+j}|\mathbf{w}_t)$$

- $L(\mathbf{w})$ is the likelihood.

The model

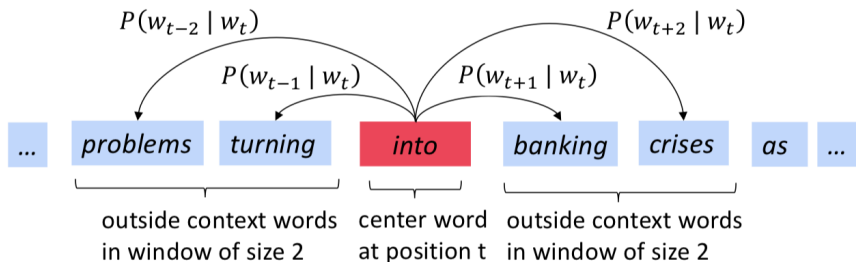
- Loss function is the negative log likelihood:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{T} \log L(\mathbf{w}) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m; j \neq 0} \log p(\mathbf{w}_{t+j} | \mathbf{w}_t)$$

- *Minimizing the loss is equivalent to maximizing the likelihood.*
- How to calculate $p(\mathbf{w}_{t+j} | \mathbf{w}_t)$? Use two vectors for each word:
 - ▶ v_w when w is a center word
 - ▶ u_w when w is a context word
 - ▶ It is not likely that a word occurs in its own context
- Use the **softmax** (generalization of the sigmoid) to predict the probabilities of a c (center), o (context) pair:

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Example



We learn to predict:

- $p(u_{problems} | v_{into})$
- $p(u_{turning} | v_{into})$
- $p(u_{banking} | v_{into})$
- $p(u_{crises} | v_{into})$
- ...

Credit: Stanford CS224N.

Softmax

Exponentiation makes anything positive

Dot product compares similarity of o and c .
 $u^T v = u \cdot v = \sum_{i=1}^n u_i v_i$
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Normalize over entire vocabulary to give probability distribution

- The softmax maps any value to a probability distribution.
- It amplifies large values (*max*) but still gives non-zero probabilities to small values (*soft*).

Credit: Stanford CS224N.

Training via SGD

- Parameters: our word embeddings, **two per word**.
- Usually, these vectors have length d within 50-1000, thus $d \ll |V|$.
- Use gradient descent to optimize and find a minimum of the loss.

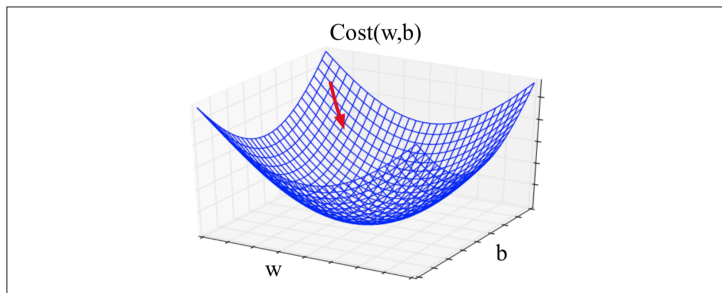


Figure 5.4 Visualization of the gradient vector in two dimensions w and b .

Credit: J&M, ch. 5.

Training via SGD

- Let us ignore for a moment the normalization term $\frac{1}{T}$ and the external summations, which are straightforward.
- Let us take the first (partial) derivative w.r.t. v_c (similarly, you can do this for u_o):

$$\begin{aligned}\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} &= u_o - \sum_{x \in V} \frac{\exp(u_x^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot u_x \\ &= u_o - \sum_{x \in V} p(x|c) \cdot u_x\end{aligned}$$

- Thus the derivative w.r.t. the central word vector v_c is the vector for the current context word u_o (true value of the context vector), minus the weighted average of the model's current representations of other possible contexts (predicted values for the context vector).
- $u_x =$ hidden layer

Optimization

- Summing over entire vocabulary for each gradient descent update is computationally expensive
- Subsampling frequent words
 - ▶ Word pair “the”, “fox” is not very informative
 - ▶ Delete highly frequent words from training text with a probability depending on their frequency
- Negative Sampling
 - ▶ Instead of adjusting all weights not occurring in the context of a word, only adjust some of them
 - ▶ Randomly select according to unigram probability

Extended SGNS calculation example: https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

Final words on Word2Vec

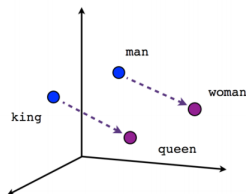
- After having trained the model, we typically use the vectors v_w or the average of v_w and u_w . We use two vectors as a kind of trick to make the derivation (and thus training) simpler.
- What we discussed is called the **Skip-gram** model.
- Alternatively, we can predict the center word using the context words: this is called the Continuous Bag Of Words model (**CBOW**).
- Note that the training objective was to predict context words (or to predict the center word), however, this was not the task
 - ▶ We just use it as a method for obtaining word vectors
- Thus, we cannot evaluate just by seeing how well the model achieves the training objective

Hyperparameters

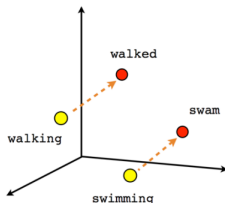
Choices when training a model

- Dimension size
- Window size
- Architecture (Skipgram/CBOW)
- Epochs
- Learning rate
- Subsampling - frequency limit for handling higher frequency items
- Negative sampling - how many 'negative' words to update

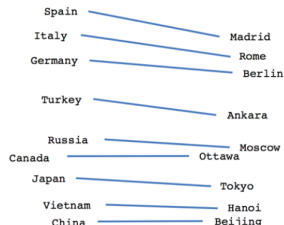
Analogy evaluation



Male-Female



Verb tense



Country-Capital

- Word2Vec became famous by being able to solve analogies by vector arithmetic
- $V_{queen} = V_{king} - V_{man} + V_{woman}$
- $\frac{p(w|a)}{p(w|b)} \approx \frac{p(w|x)}{p(w|y)}$

http://epsilon-it.utu.fi/wv_demo/

Variants

- Doc2Vec
 - ▶ Numeric representation of documents, rather than words
 - ▶ Word2Vec, but also add document ID as feature vector for prediction
 - ▶ Obtain document vectors for e.g. information retrieval
- Sent2Vec
 - ▶ Numeric representation of sentences, rather than words
 - ▶ Average of Word2Vec word vectors for a sentence
 - ★ Includes n-gram embeddings rather than just unigram embeddings
- Nonce2Vec
 - ▶ Extension of Word2Vec for training unknown/low-frequency words into an existing Word2Vec model
 - ▶ Incremental learning: more cognitively plausible
 - ▶ Includes parameter decay, e.g. a high learning rate for the first example which decreases for further examples

So far

- So far we have seen count-based approaches to word vectors:
 - ① Make use of corpus statistics.
 - ② Very fast training (just count..).
 - ③ *Sensitive to large counts.*
 - ④ *Mostly only capture word similarity.*
- And approaches based on prediction tasks (Word2Vec):
 - ① Can capture more complex patterns.
 - ② Generate better performance as features for other tasks.
 - ③ *Do not make use of corpus statistics.*

GloVe: Intuition

- **GloVe key idea: capture ratios of co-occurrence probabilities as linear meaning components in a vector space.**
- Estimated over a whole corpus

	$x = \text{solid}$	$x = \text{gas}$	$x = \text{water}$	$x = \text{random}$
$P(x \text{ice})$	large	small	large	small
$P(x \text{steam})$	small	large	large	small
$\frac{P(x \text{ice})}{P(x \text{steam})}$	large	small	~ 1	~ 1

Credit: Stanford CS224N.

GloVe: Intuition

- **GloVe key idea: capture ratios of co-occurrence probabilities as linear meaning components in a vector space.**
- Learn a log-linear model as follows:

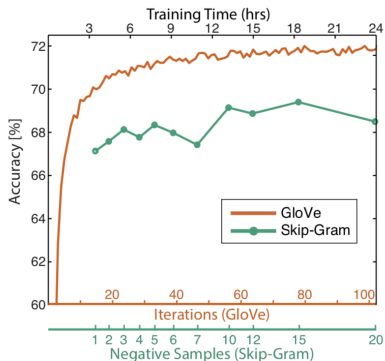
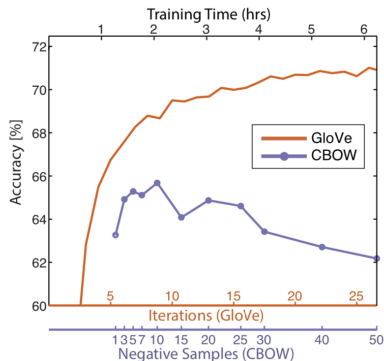
$$w_i \cdot w_j = \log(P(i|j))$$

- Able to capture vector differences for ratios:

$$w_x \cdot (w_a - w_b) = \log\left(\frac{P(x|a)}{P(x|b)}\right)$$

- Check the (excellent) paper for more details.

Comparison with Word2Vec



Credit: Stanford CS224N.

FastText: Intuition

- Use character n-grams instead of words
- N between 3 and 6
- Especially more effective for morphologically rich languages such as Arabic, Russian, Turkish
- Addresses out-of-vocabulary problem
- Better at syntactic tasks

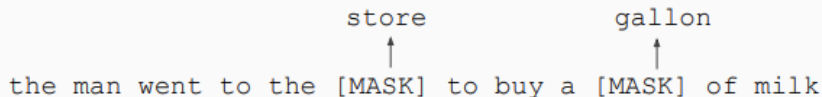
Contextual Word Embeddings: Intuition

More details in Week 8 (we need to learn about recurrent architectures and transformer models first)

- Models learn to generate a vector for a word based on its context
- Considers position of the word in the sentence
- Subtokenization
- Sentence embeddings
- Can obtain static embeddings from input layer
- Add and tune different output layers to perform different tasks, such as a sentence classifier or token classifier

BERT

- Training objective 1: Mask out $k\%$ of input words and predict the masked words
- Training objective 2: Next sentence prediction (actual or random)



store gallon

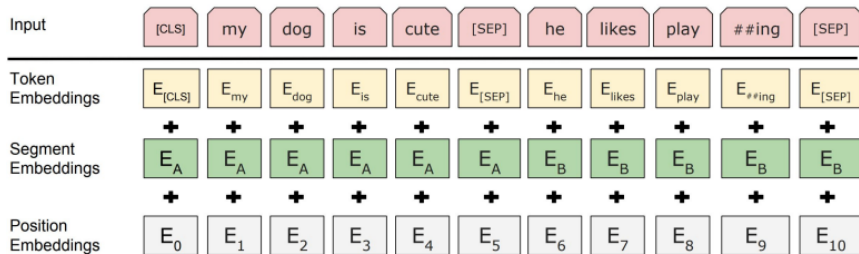
↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

Credit: Stanford CS224N.

BERT

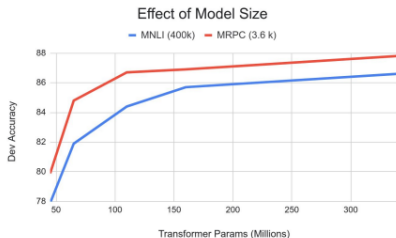
- 30.000 word vocabulary of subtokens
- Each token is sum of three embeddings



Credit: Stanford CS224N.

BERT

- Bigger is better...



Credit: Stanford CS224N.

References

- Stanford CS224N classes 1 and 2:
<http://web.stanford.edu/class/cs224n/index.html>.
- Good tutorial <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model>.
- Original Word2Vec paper
<https://arxiv.org/pdf/1301.3781.pdf>.
- Negative sampling paper <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-pdf>.
- GloVe <https://nlp.stanford.edu/pubs/glove.pdf>.
- Evaluation of word embeddings:
<https://www.aclweb.org/anthology/D15-1036>.

Note: there is much more. Ask me if you are interested.

Word2Vec Expanded (optional)

Derivation for softmax

- First, we need some notable derivatives:

$$\frac{\partial \log(x)}{\partial x} = \frac{1}{x}$$

$$\frac{\partial \exp(x)}{\partial x} = \exp(x)$$

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} \rightarrow \text{chain rule}$$

Derivation for softmax

- We can divide in two parts:

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} = \frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) - \frac{\partial}{\partial v_c} \log \sum_{w \in V} \exp(u_w^T v_c)$$

- First part:

$$\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) = u_o$$

- Second part:

$$\begin{aligned} \frac{\partial}{\partial v_c} \log \sum_{w \in V} \exp(u_w^T v_c) &= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot \frac{\partial}{\partial v_c} \sum_{x \in V} \exp(u_x^T v_c) \\ &= \frac{\sum_{x \in V} \exp(u_x^T v_c) u_x}{\sum_{w \in V} \exp(u_w^T v_c)} \end{aligned}$$

Derivation for softmax

- Combine:

$$\begin{aligned}\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} &= u_o - \frac{\sum_{x \in V} \exp(u_x^T v_c) u_x}{\sum_{w \in V} \exp(u_w^T v_c)} \\ &= u_o - \sum_{x \in V} p(x|c) u_x\end{aligned}$$

Noise-contrastive estimation

Exponentiation makes anything positive

Dot product compares similarity of o and c .

$$u^T v = u \cdot v = \sum_{i=1}^n u_i v_i$$

Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Normalize over entire vocabulary
to give probability distribution

- Normalizing over the entire vocabulary is very expensive.
- Idea: let us just **sample some negative examples** (collocations absent in the data), and train a **binary logistic regression classifier** to distinguish between positive (real) and negative (fake) pairs.
- For every center-context pair, also sample K negative pairs. The center-context pair is going to be a positive datapoint, the negative pairs are negative datapoints.
- The logistic classifier then uses the same dot product of vectors as features, and a cross-entropy loss.