Word Embeddings

Jelke Bloem & Giovanni Colavizza

Text Mining
Amsterdam University College

March 4, 2024

Assignments

- Assignment 2 (word embeddings): 15/03
- Reading assignment Word2Vec: 08/03

Questions/discussion reading assignment 1

- How could we overcome bias and misinformation in NLP models?
- What are the currently employed approaches and strategies for preventing the bias in language models? Do the biases that are so prevalent in the training data have to be manually adjusted, or are there any more efficient ways to "neutralise" the models? If the task of correcting bias is not incentivised, how can we prevent the models from cementing the biases into our systems and services?
- Should the field of NLP be focusing on trying to create better novel approaches or on maxing out / adjusting the current solutions and testing them out in different context?

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?
- How could one combine the statistical with the theoretical approach to NLP? Would this be possible with word vectors?
- What requires a language model/technology to ever achieve a full understanding of human language?
- Discuss the relationship that Chomsky establishes between Plato's concepts of the ideal forms, the immortal soul and analogy of the cave and his own views on the innate structures of language? What implications does this relationship have for our understanding of language learning and processing in both humans and machines?

Overview

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     piano   together]…
                    …The [first electric     pianos  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…
…mary 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

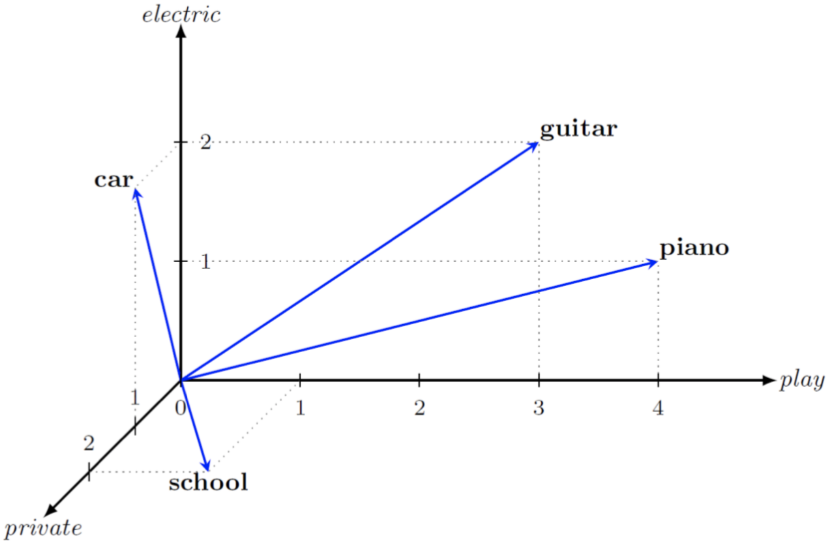
- We have a set of words V and a set of contexts they occur into C, taken from our corpus of documents. X in this case is a $|V| \times |C|$ matrix with word occurrences in contexts.
- The most intuitive context are co-occurrences with other words in V, within a certain **window**. In this case, X would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

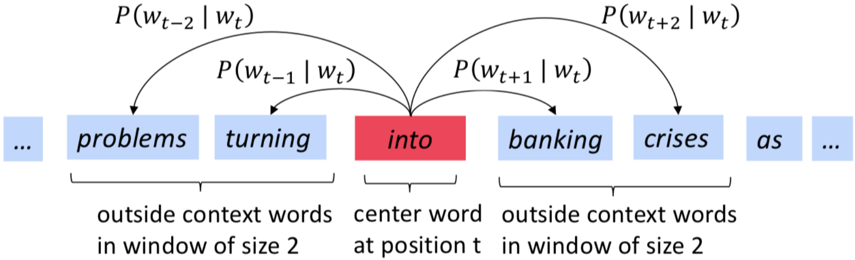
- **Sparse vectors**: many zero values and high-dimensional spaces. E.g., weighted co-occurrence matrices.
- **Dense vectors**: no zero values and comparatively smaller-dimensional spaces.
  - Dimensionality reduction (Singular Value Decomposition, Random indexing, Non-negative matrix factorization).
  - **Neural-network inspired** (Word2Vec, GloVe, BERT and many more): we start today.

Word2Vec

Intuition

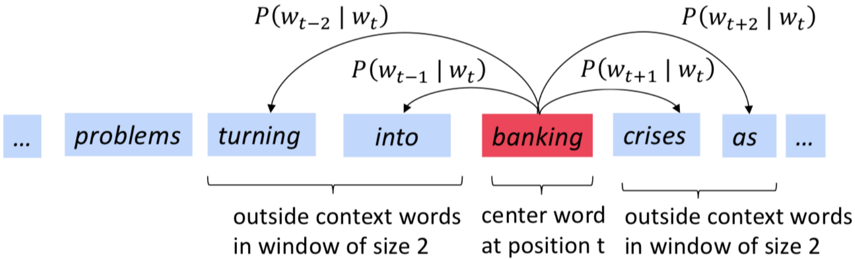
- **Word2Vec**: a framework for learning dense word vectors.
- Idea:
  1. We have a large corpus of text.
  2. We want each word in the vocabulary to be represented by a vector.
  3. We can go through the corpus and establish a *context o* for every *center/focus word c*, using a certain window/span.
  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



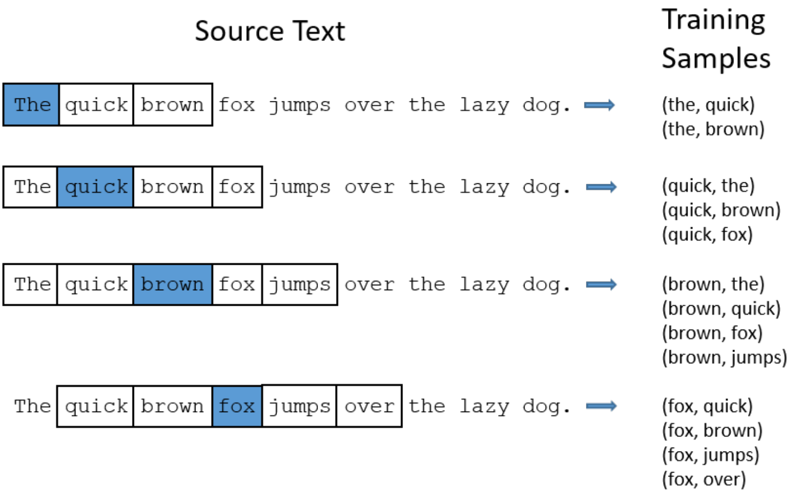
Credit: Stanford CS224N.

Words in context



Credit: Stanford CS224N.

Words in context as data



Credit: `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):
$$L(\boldsymbol{w}) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m; j \neq 0} p(\boldsymbol{w}_{t+j}|\boldsymbol{w}_t)$$
- $L(\boldsymbol{w})$ is the likelihood.

The model

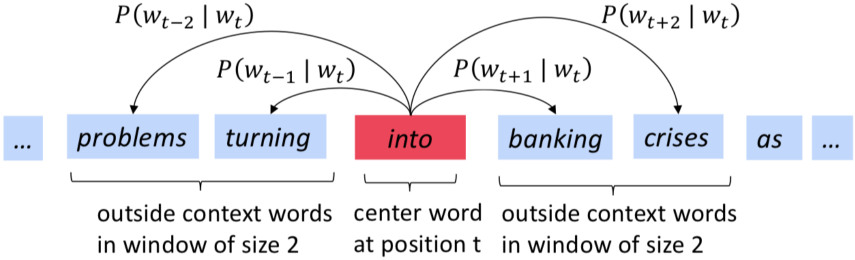
- Loss function is the negative log likelihood:

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

- *Minimizing the loss is equivalent to maximizing the likelihood.*
- How to calculate $p(\boldsymbol{w}_{t+j} | \boldsymbol{w}_t)$? 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

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

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

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.

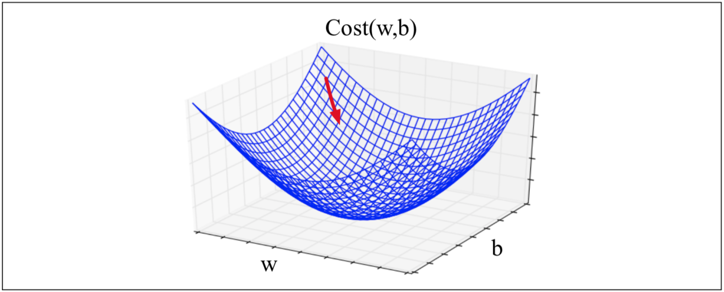


Figure 5.4 Visualization of the gradient vector in two dimensions *w* and *b*.

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

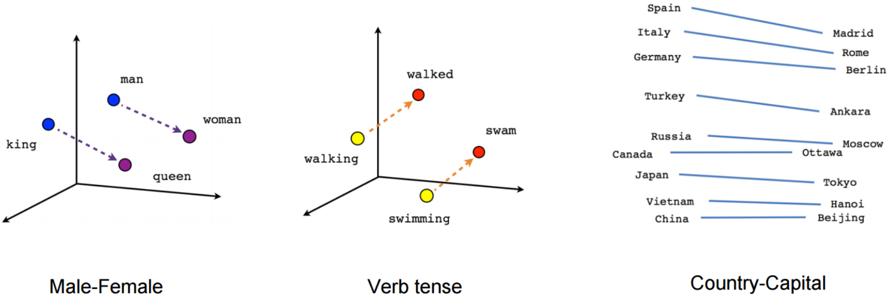
- After having trained the model, we typically use the vectors v_w or the average of v_w and u_w. We use two vectors as a kind of trick to make the derivation (and thus training) simpler.
- What we discussed is called the **Skip-gram** model.
- Alternatively, we can predict the center word using the 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



- 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

GloVe

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.*

Intuition

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

	x = solid	x = gas	x = water	x = 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.

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:

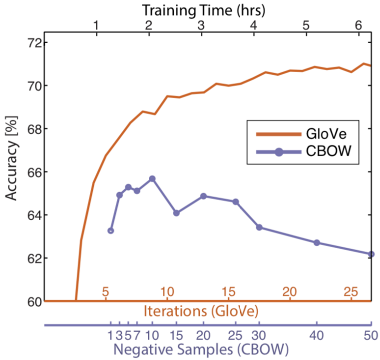
$$w_i \cdot w_j = log\big(P(i|j)\big)$$

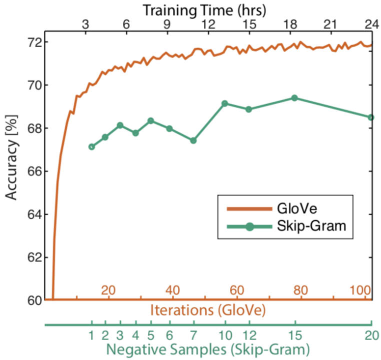
- Able to capture vector differences for ratios:

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

- Check the (excellent) paper for more details.

Comparison with Word2Vec





Credit: Stanford CS224N.

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:

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

- First part:

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

- Second part:

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

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

Derivation for softmax

- Combine:

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

$$= u_o - \sum_{x \in V} p(x|c) u_x$$

Noise-contrastive estimation

Exponentiation makes anything positive

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

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

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.

Credit: Stanford CS224N.