Deep Learning for NLP

Week 5: Word Embeddings 2

Sharid Loáiciga — May 19th, 2020

Agenda

- Lexical vs vector semantics
- Embeddings
    - Counts-based
    - Dense
    - word2vec
- Embeddings flavors:
    - debiased
    - universal
    - ELMO & BERT

Lexical semantics:

the study of word meaning

lemma ≠ wordforms (mouse, mice)

```
mouse(N)
 sense1 any of numerous small rodents
 sense2 a hand-operated device that controls a cursor
```

Words can have several meanings or senses and this contributes to ambiguity. Theses senses have different relationships:

synonymy: couch-sofa, car-automobile, water-H2O
word similarity: cats-dogs, bathroom-kitchen
connotation: reflect of a person's emotions, great-love, terrible-hate

Words don't have many synonyms
but most have lots of similar words

Early work on connotations

Osgood et al., 1957: words vary along three important dimensions of affective meaning:

- valence: the pleasantness of the stimulus
- arousal: the intensity of emotion provoked by the stimulus
- dominance: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

heartbreak

can be represented as the point

[2.45,5.65,3.58]

Vector semantics

How do we capture all those nuances of meaning?

- Joos (1950), Harris (1954) &Firth (1957) proposed to use word **distributions**: the set of contexts in which each word occurs

1 *Ongchoi* is delicious sauteed with garlic.
2 *Ongchoi* is superb over rice.
3 ... *ongchoi* leaves with salty sauces...

4 ... spinach sauteed with garlic over rice...
5 ...chard stems and leaves are delicious...
6 ...collard greens and other salty leafy greens

- The best current models are vector-based

Goal: to represent a word as a point in some multidimensional semantic space. These vectors are called **embeddings**, because the word is embedded in a particular vector space.

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015).

Two birds: the distributionalist intuition (defining a word by counting what other words occur in its environment), and the vector intuition of Osgood et al. (1957)

Count-based embeddings

context ±4

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

term x term matrix

$|V| \times |V|$

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purp d in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

tel:3325%203982%20378%205%2013

Count-based embeddings

- Not all words are the same. Raw frequency is very skewed and not very discriminative. For example, maybe *pie* is nearby *cherry*, as is *the* or *good* which are even more frequent but unimportant.

- Pointwise Mutual Information (PMI): how much more the two words co-occur in our corpus than we would have a priori expected them to appear by chance

$$\text{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

$P(w, c)$ → how often we observe a co-occurence

$P(w)P(c)$ → how often we expect it

Count-based embeddings

- |V| is generally the size of the vocabulary, usually between 10,000 and 50,000 words But most of these numbers are zero, making these vector representations **sparse** & **long**.

- There are algorithms for taking care of this, ex., singular value decomposition, although complexity can be high.

Dense embeddings

- **short** (of length perhaps 50-1000) & **dense** (most values are non-zero)

- originated with a **language modelling approach**

Digression

Language model

- Goal: determine $P(s = w_1 \dots w_k)$ in some domain of interest

$$P(s) = \prod_{i=1}^{k} P\left(w_i \mid w_1 \dots w_{i-1}\right)$$

$$\text{e.g., } P\left(w_1 w_2 w_3\right) = P\left(w_1\right) P\left(w_2 \mid w_1\right) P\left(w_3 \mid w_1 w_2\right)$$

- Traditional n-gram language model assumption:
  "the probability of a word depends only on **context** of $n-1$ previous words"

$$\Rightarrow \widehat{P}(s) = \prod_{i=1}^{k} P\left(w_i \mid w_{i-n+1} \dots w_{i-1}\right)$$

- Typical ML-smoothing learning process (e.g., Katz 1987):
  1. compute $\widehat{P}\left(w_i \mid w_{i-n+1} \dots w_{i-1}\right) = \frac{\#w_{i-n+1} \dots w_{i-1} w_i}{\#w_{i-n+1} \dots w_{i-1}}$ on training corpus
  2. smooth to avoid zero probabilities

Slide credit: Antoine Tixier

Digression

Traditional n-gram language model

Limitation 1): curse of dimensionality

- Example
- train a 10-gram LM on a corpus of 100.000 unique words
- space: 10-dimensional hypercube where each dimension has 100.000 slots
- model training ↔ assigning a probability to each of the 100.000^{10} slots
- **probability mass vanishes** → more data is needed to fill the huge space
- the more data, the more unique words! → vicious circle
- what about corpuses of 10^6 unique words?

- → in practice, contexts are typically limited to size 2 (trigram model)
  e.g., famous Katz (1987) smoothed trigram model

- → such short context length is a limitation: a lot of information is not captured

Slide credit: Antoine Tixier

Digression

Traditional n-gram language model

Limitation 2): word similarity ignorance

- We should assign similar probabilities to `Obama speaks to the media in Illinois` and `the President addresses the press in Chicago`

- This does not happen because of the "one-hot" vector space representation:

$$\left.\begin{aligned} \text{obama} &= [0\ 0\ 0\ 0\ \ldots\ 0\ 1\ 0\ 0] \\ \text{president} &= [0\ 0\ 0\ 1\ \ldots\ 0\ 0\ 0\ 0] \end{aligned}\right\} \quad \overrightarrow{\text{obama}}.\overrightarrow{\text{president}} = \vec{0}$$

$$\left.\begin{aligned} \text{speaks} &= [0\ 0\ 1\ 0\ \ldots\ 0\ 0\ 0\ 0] \\ \text{addresses} &= [0\ 0\ 0\ 0\ \ldots\ 0\ 0\ 1\ 0] \end{aligned}\right\} \quad \overrightarrow{\text{speaks}}.\overrightarrow{\text{addresses}} = \vec{0}$$

$$\left.\begin{aligned} \text{illinois} &= [1\ 0\ 0\ 0\ \ldots\ 0\ 0\ 0\ 0] \\ \text{chicago} &= [0\ 1\ 0\ 0\ \ldots\ 0\ 0\ 0\ 0] \end{aligned}\right\} \quad \overrightarrow{\text{illinois}}.\overrightarrow{\text{chicago}} = \vec{0}$$

- In each case, word pairs share no similarity

- We need to encode **word similarity** to be able to **generalize**

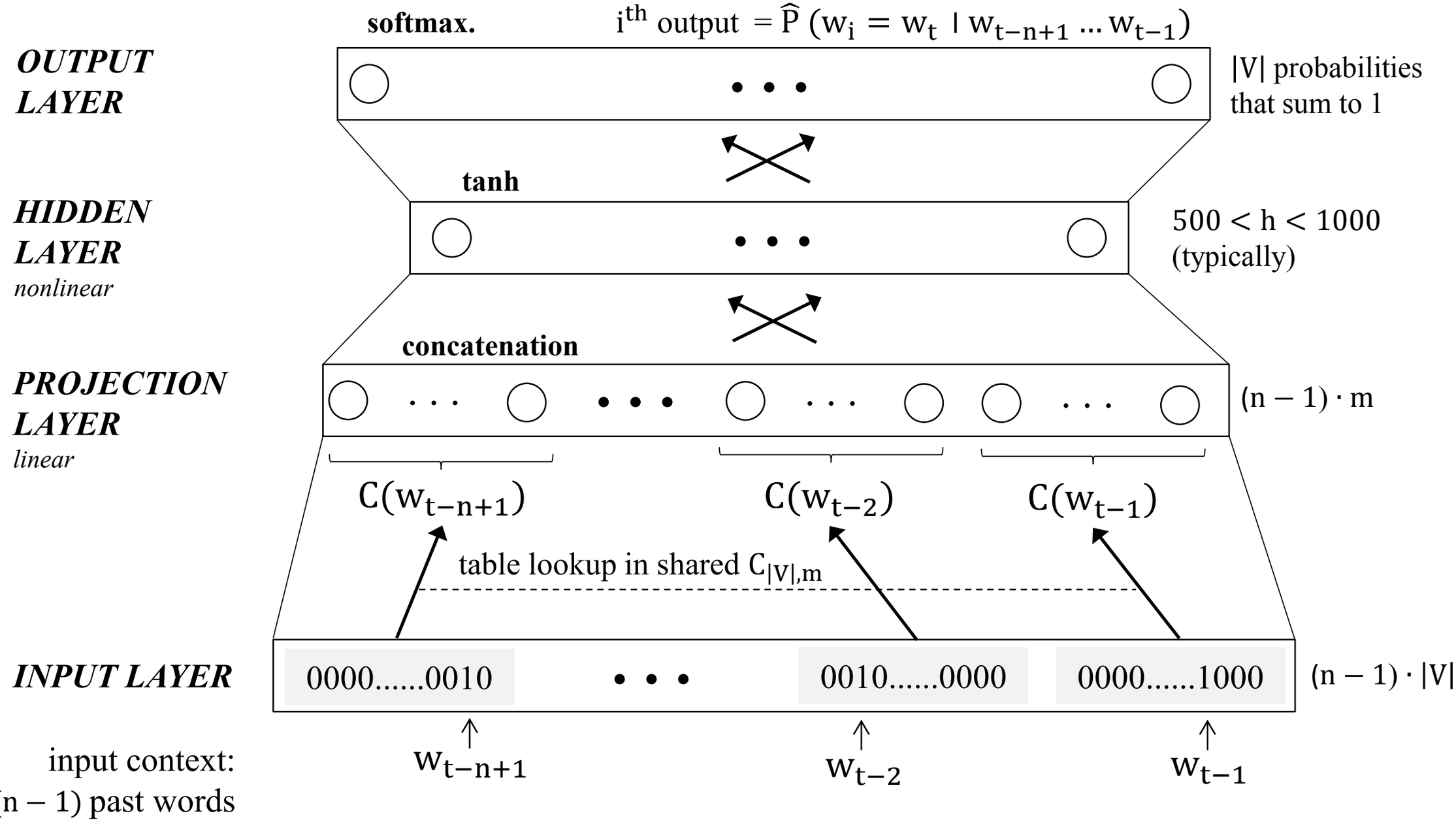
Slide credit: Antoine Tixier

Digression

Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \ldots w_{t-1}, w_t)$

objective: minimize $E = -\log \hat{P}(w_t \mid w_{t-n+1} \ldots w_{t-1})$

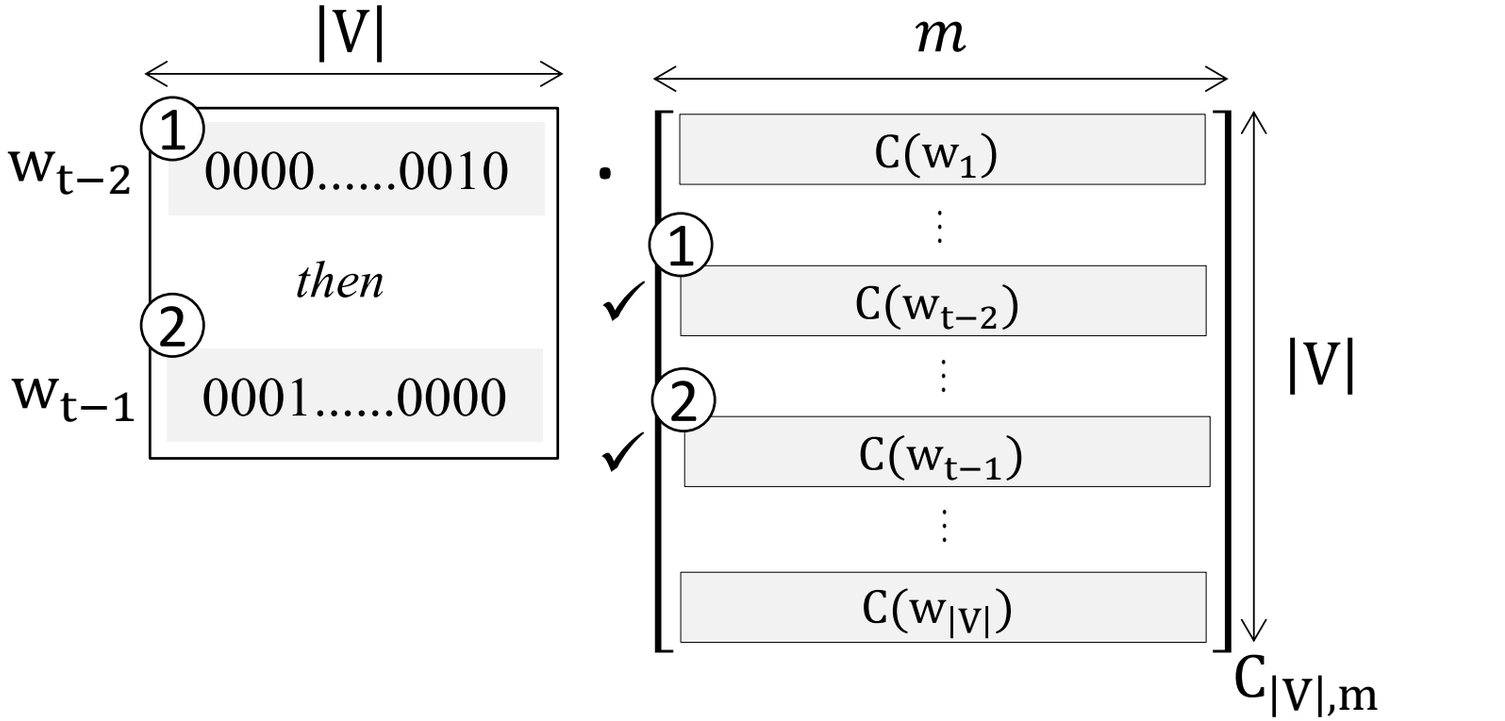


Slide credit: Antoine Tixier

Digression NNLM Projection layer

- Performs a simple table lookup in $C_{|V|,m}$: concatenate the rows of the shared mapping matrix $C_{|V|,m}$ corresponding to the context words

Example for a two-word context $w_{t-2}w_{t-1}$:



Concatenate ① and ② → $C(w_{t-2})$ $C(w_{t-1})$

- $C_{|V|,m}$ is **critical**: it contains the weights that are tuned at each step. After training, it contains what we're interested in: the **word vectors**

Slide credit: Antoine Tixier

Digression NNLM hidden/output layers and training

- Softmax (log-linear classification model) is used to output positive numbers that sum to one (a multinomial probability distribution):

  for the i^{th} unit in the output layer: $\hat{P}(w_i = w_t \mid w_{t-n+1} \dots w_{t-1}) = \frac{e^{y w_i}}{\sum_{i'=1}^{|V|} e^{y w_{i'}}}$

  Where:
  - $y = b + U.\tanh(d + H.x)$
  - tanh : nonlinear squashing (link) function
  - x : concatenation $C(w)$ of the context weight vectors seen previously
  - b : output layer biases ($|V|$ elements)
  - d : hidden layer biases (h elements). Typically $500 < h < 1000$
  - U : $|V| * h$ matrix storing the *hidden-to-output* weights
  - H : $(h * (n-1)m)$ matrix storing the *projection-to-hidden* weights

  $\rightarrow \boldsymbol{\theta} = (\mathbf{b}, \mathbf{d}, \mathbf{U}, \mathbf{H}, \mathbf{C})$

- Complexity per training sequence: $n * m + n * m * h + \mathbf{h} * \mathbf{|V|}$

  computational bottleneck: **nonlinear hidden layer** ($h * |V|$ term)

- **Training** is performed via stochastic gradient descent (learning rate ε):

$$\theta \leftarrow \theta + \varepsilon \cdot \frac{\partial E}{\partial \theta} = \theta + \varepsilon \cdot \frac{\partial \log \hat{P}\,(w_t \mid w_{t-n+1} \dots w_{t-1})}{\partial \theta}$$

  (weights are initialized randomly, then updated via backpropagation)

word2vec (Google)

Mikolov et al. (2013a)

- Two algorithms:
    - CBOW: from context predict target
    - Skip-gram: from target predict context
      (focus from here)
- **GloVe** (Stanford): based on word-to-word co-occurrence
- **fastText** (Facebook): uses character-level representations

Skip-gram with negative sampling

- Train a classifier on a binary prediction task: “Is word *w* likely to show up near *apricot*?” If yes, this is the gold ‘correct answer’.

- We don’t actually care about this prediction task; instead we’ll take the learned classifier weights as the word embeddings.

- much simpler model than the neural network language model:
  - **simplifies the task:** binary classification instead of word prediction
  - **simplifies the architecture:** logistic regression classifier instead of a multi-layer neural network with hidden layers

Learning Skip-gram embeddings

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    t     c3       c4
```

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Given the set of positive and negative training instances, and an initial set of embeddings, the goal of the learning algorithm is to adjust those embeddings such that we

- Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive examples
- Minimize the similarity of the (t,c) pairs drawn from the negative examples.

We can express this formally over the whole training set as:

$$L(\theta) = \sum_{(t,c)\in+} \log P(+|t,c) + \sum_{(t,c)\in-} \log P(-|t,c) \tag{6.34}$$

Learning Skip-gram embeddings

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    t     c3       c4
```

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

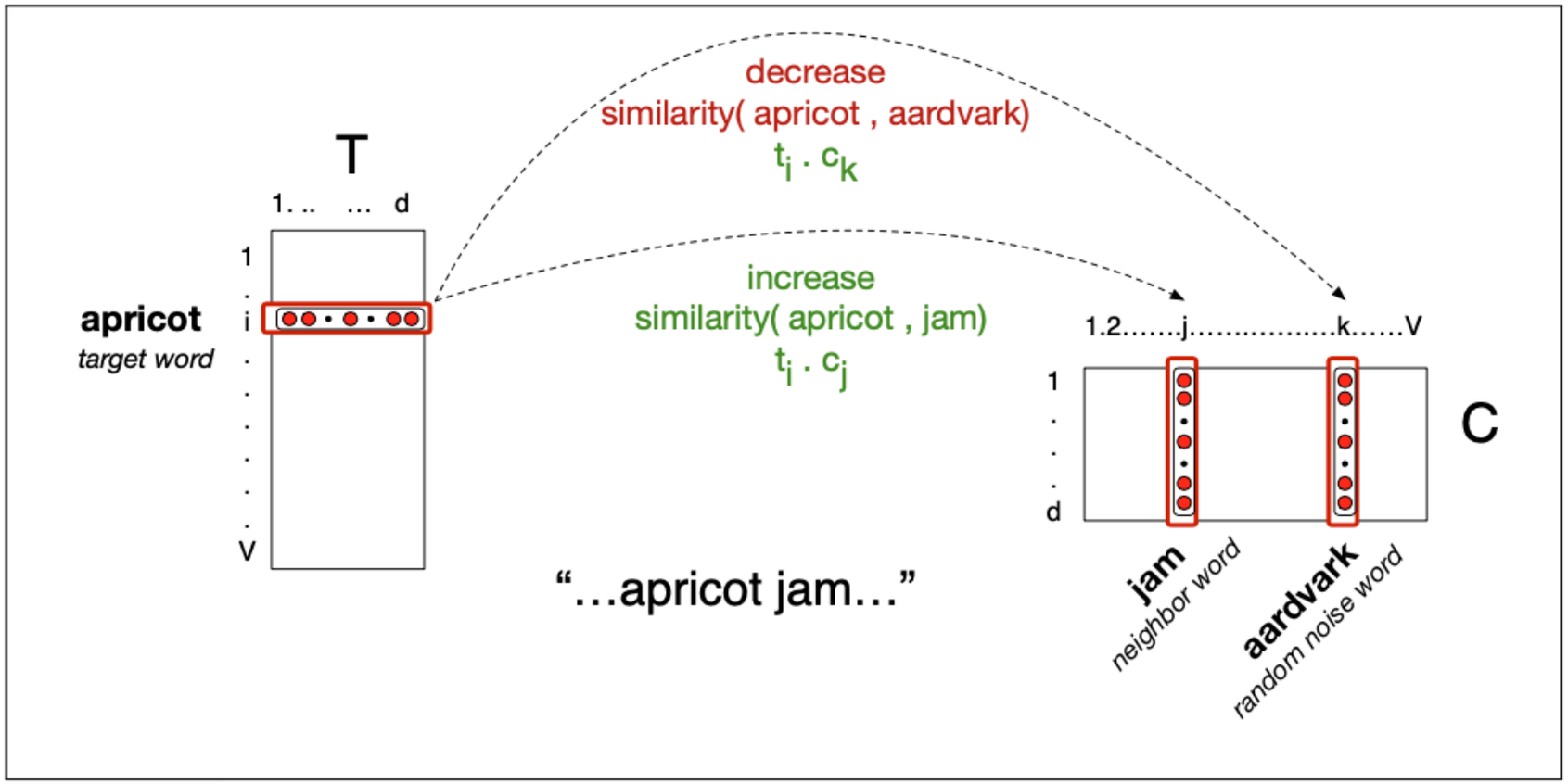


Figure 6.12 The skip-gram model tries to shift embeddings so the target embeddings (here for *apricot*) are closer to (have a higher dot product with) context embeddings for nearby words (here *jam*) and further from (have a lower dot product with) context embeddings for words that don't occur nearby (here *aardvark*).

Pre-trained embeddings are normal in NLP

Typically you train word embeddings on huge datasets:

Google 300 dimensional word embeddings trained on 100 billion tokens from GoogleNews.

This allows us to derive knowledge about words which do not occur in our own task-specific dataset.

We get knowledge about “awesome” because its embedding will be similar to “great”.

Embeddings are cool

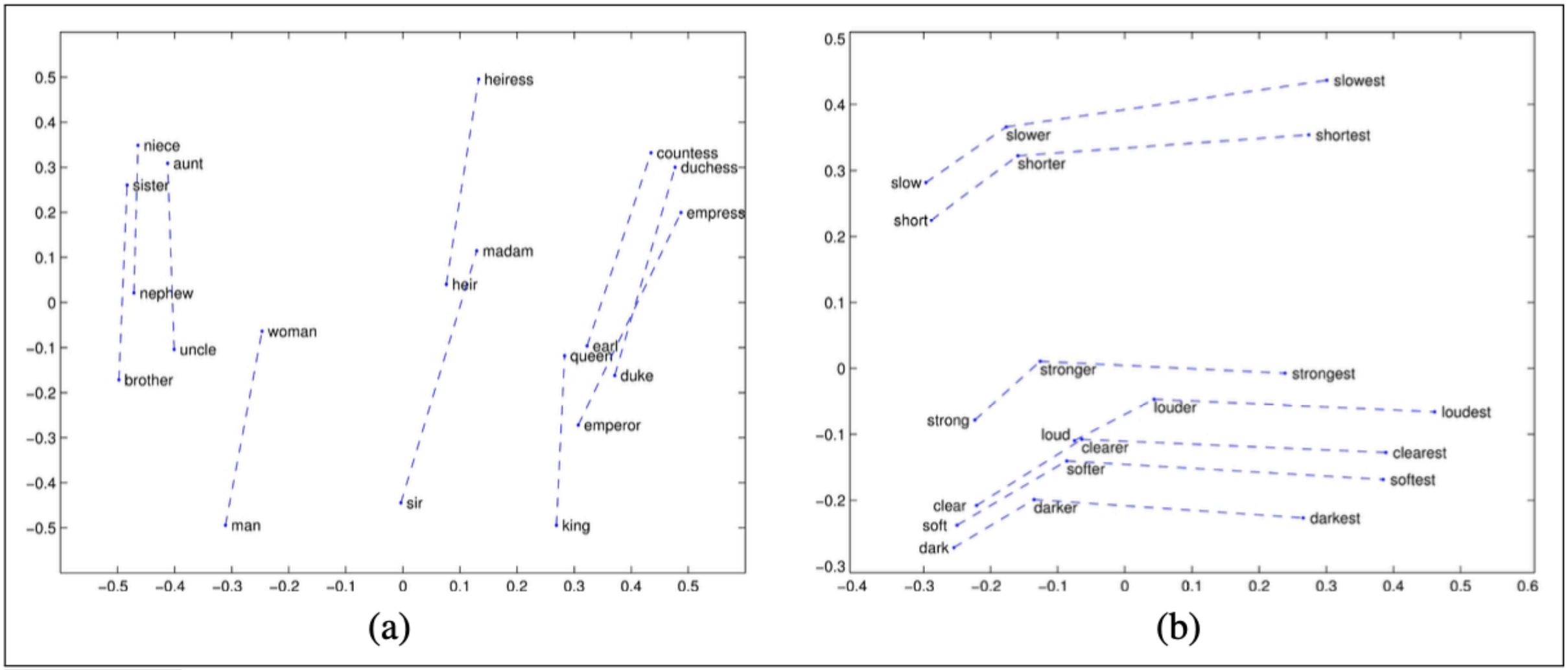


Figure 6.13 Relational properties of the vector space, shown by projecting vectors onto two dimensions. (a) 'king' - 'man' + 'woman' is close to 'queen' (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

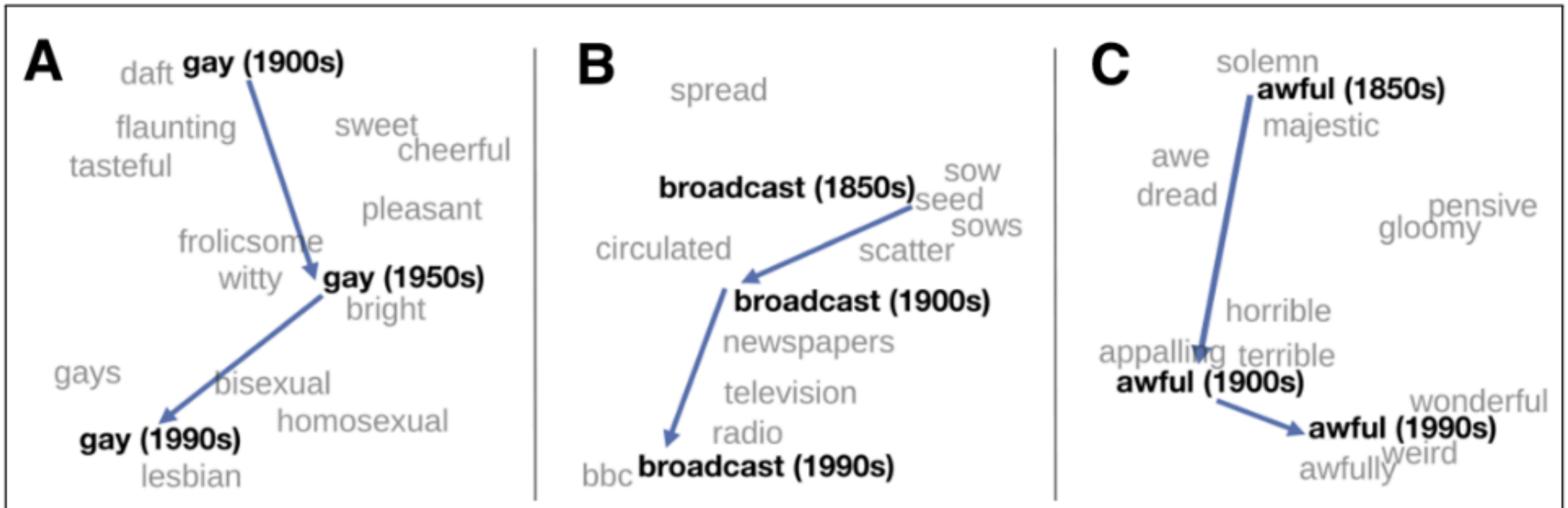


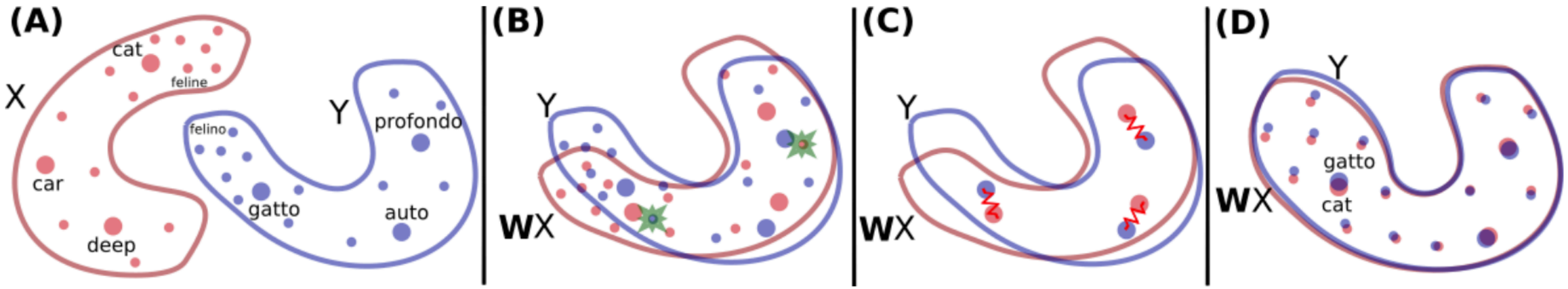
Figure 6.14 A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors. The modern sense of each word, and the grey context words, are computed from the most recent (modern) time-point embedding space. Earlier points are computed from earlier historical embedding spaces. The visualizations show the changes in the word *gay* from meanings related to "cheerful" or "frolicsome" to referring to homosexuality, the development of the modern "transmission" sense of *broadcast* from its original sense of sowing seeds, and the pejoration of the word *awful* as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Hamilton et al., 2016).

…but are dependent on data

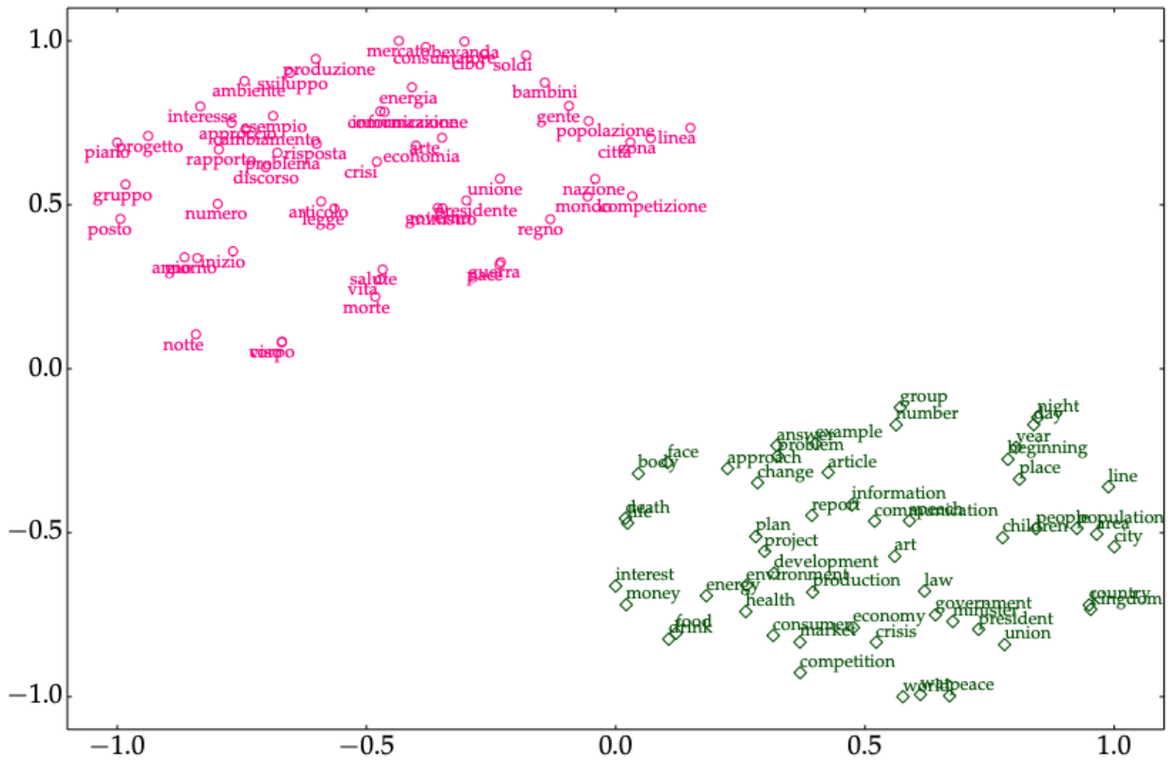
- man : woman :: king : queen ?
- flowers : pleasantness :: insects : unpleasantness ??
- father : doctor :: mother : nurse ???
- man : computer-programmer :: woman : homemaker ???

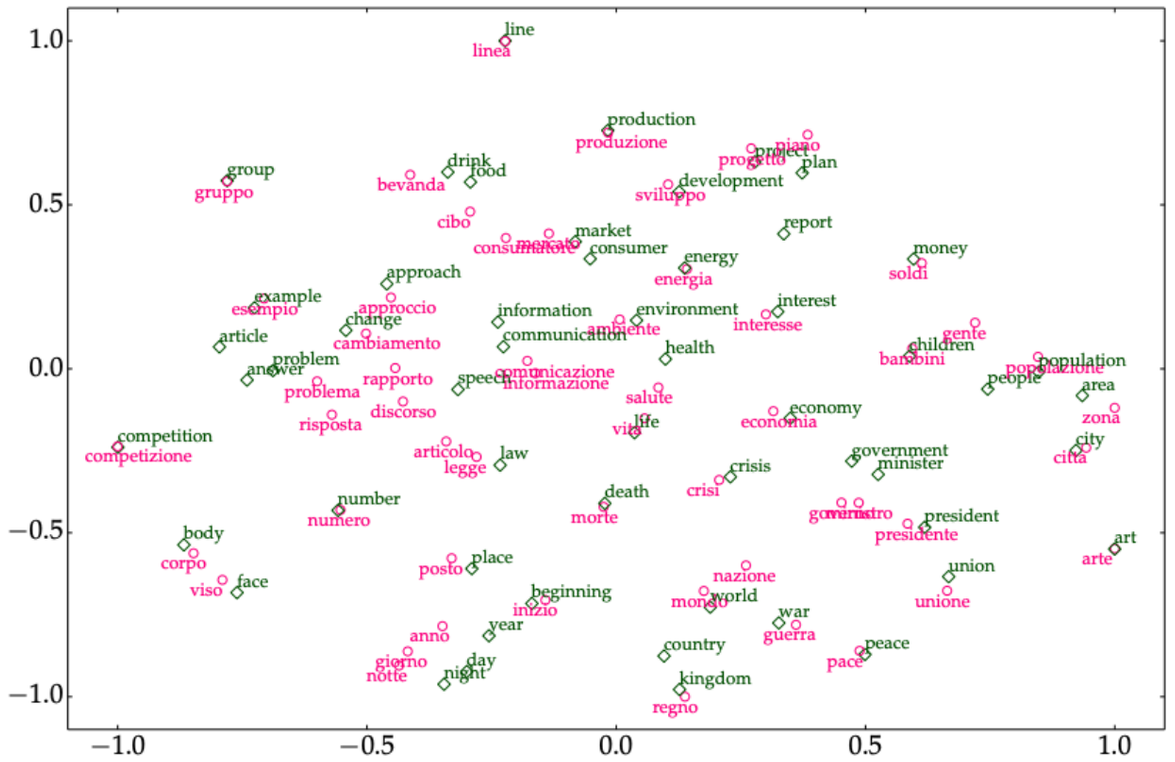
Bias is still an open problem:
(Bolukbasi et al. 2016)
(Zhao et al. 2017)
(Gonen and Goldberg, 2019)

Multilingual embeddings



https://github.com/facebookresearch/MUSE





https://arxiv.org/pdf/1706.04902.pdf

Contextualized word vectors

Elmo (Peters et al. 2018)

https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

BERT (Peters et al. 2018)

https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/

References

- Bengio, Y., Ducharme, R., Vincent P. & Jauvin, C. (2003). A Neural Probabilistic Language Model. Journal of Machine Learning Research.

- Bolukbasi et al. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. NIPS.

- Devlin J. et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.

- Gonen, H. & Goldberg, Y. (2019). Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them. NAACL.

- Jurafsky, D. & Martin, J.H. (2020). Speech and Language Processing (3rd ed. draft) https://web.stanford.edu/~jurafsky/slp3/

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

- Peters M., et al . (2018). Deep Contextualized Word Representations. NAACL.

- Zhao, et al. (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints. EMNLP.