Word Embeddings (Word2Vec)

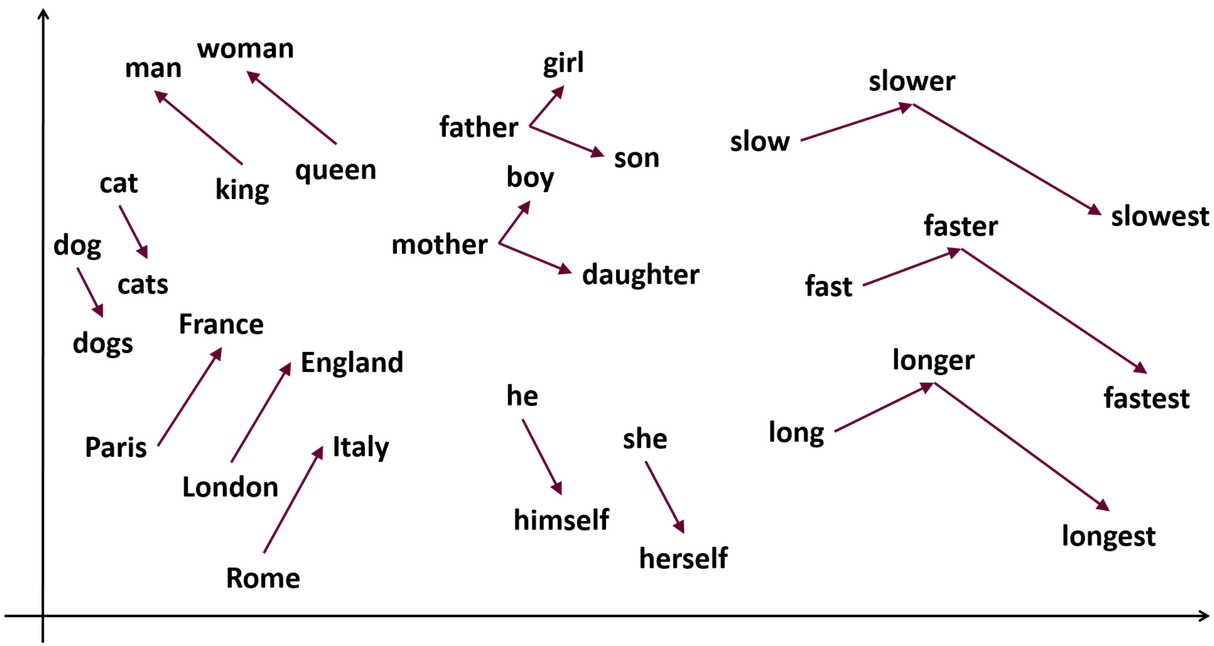
David I. Inouye

Fundamental question: How should **words** be represented so that relationships are encoded?

- Let w_i denote the representation of the i-th word
- Synonyms should be similar
  - "Car" and "auto", e.g., $\text{sim}(w_{car}, w_{automobile}) = 1$
- Antonyms should be dissimilar
  - "Good" and "bad", e.g., $\text{sim}(w_{good}, w_{bad}) = -1$
- Related words would have some similarity
  - "School" and "book", e.g., $\text{sim}(w_{school}, w_{book}) = 0.5$

Fundamental question: How should <u>**words**</u> be represented so that relationships are encoded?

- Encode more complex relationships like analogies
  - King is to man as queen is to ____.



https://samyzaf.com/ML/nlp/nlp.html

<u>Word embeddings</u> can be used to represent words with dense vectors that encode semantic relationships

- Would one-hot vectors work?
  - i.e., $w_1 = [1,0,0,\cdots], w_2 = [0,1,0,\cdots], w_3 = [0,0,1,\cdots],$

- Goal: Automatically learn **dense** word embeddings that encode **semantic information** just given a bunch of text data (easy to obtain).

- Motivation: The <u>**distributional hypothesis**</u> in linguistics assumes words that occur in the same context are related.

Word embeddings are estimated via **self-supervised learning**—a type of unsupervised learning

- **Self-supervised learning** attempts to predict part of the data given other parts
  - Predict half of an image given the other half
  - Predict future from past
  - Predict past from future
  - Our case: Predict missing word(s) in sentence
- These are “pseudo” tasks
  - After training we don’t care about the model’s predictions
  - However, latent semantic structure emerges (i.e., the word embeddings are meaningful and useful)

Word2vec Task 1: Predict middle word given words before and after target word

- Sliding window across text
  - Joe fixed the fence while working in the yard.
  - Joe fixed _____ fence while working in the yard.
  - Joe fixed the _____ while working in the yard.
  - Joe fixed the fence _____working in the yard.
  - Joe fixed the fence while _____ in the yard.

- The **target word** is the word to be predicted
- The **context** is the words before and after

Word2vec Task 1: <u>Continuous Bag-of-Words (CBOW)</u> architecture is a simple linear model

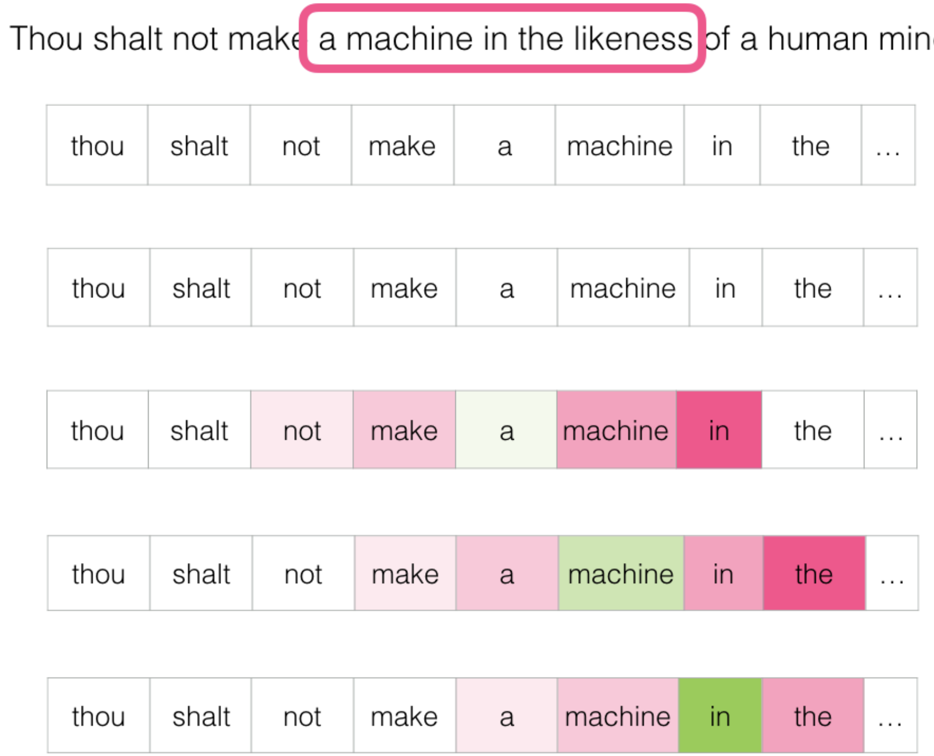
- CBOW adds the word embeddings of the context together (i.e., BoW) and then tries to predict
- $\log p(y|C) = \log \sigma(A(\sum_{i \in \mathcal{C}} w_i) + b\ )$
- The CBOW can be seen to be a

$$\log p(y|C) = \log \sigma(A(WC)\mathbf{1}_{2m} + b)$$
$$A \in \mathbb{R}^{d \times k}, \qquad W \in \mathbb{R}^{k \times d},$$
$$C \in \{0,1\}^{d \times 2m}, \qquad \mathbf{1}_{2m} = [1,1,1, \cdots]^T$$

  - Let d, k, m denote the vocabulary size, the embedding dimension and the context size

Word2vec Task 2: Predict context given target word (mirror of task 1)

- Could create paired training dataset



input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

Example from http://jalammar.github.io/illustrated-word2vec/

Word2vec Task 2: However, this task is often too computationally expensive

- Predicting one word in a large vocabulary (think millions of words) is too expensive so simplify task to predict yes or no

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

Example from http://jalammar.github.io/illustrated-word2vec/

Word2vec Task 2: **Negative sampling** of words that should not be in context is required

- Add random negative examples

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	make	1

Pick randomly from vocabulary
(random sampling)

Word	Count	Probability
aardvark		
aarhus		
aaron		
taco		
thou		
zyzzyva		

Example from http://jalammar.github.io/illustrated-word2vec/

Word2vec Task 2: The skip-gram model with negative sampling is simple

- The model merely trains a logistic regression

$$\sum_{w_c \in \mathcal{C}} \left(\log p(w_c | w_{mid}) + \sum_{w_n \sim Neg} \log(1 - p(w_n | w_{mid}) \right)$$

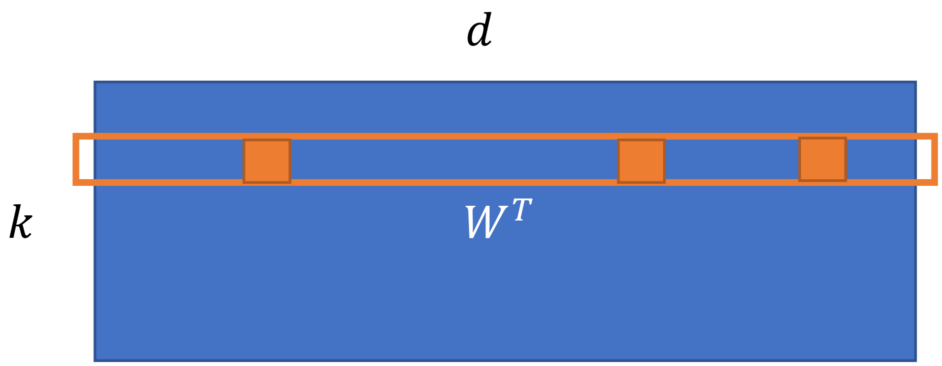
- This can be written as:

$$\sum_{w_c \in \mathcal{C}} \left(\log \sigma\left(w_c'^T w_{mid}\right) + \sum_{w_n \sim Neg} \log\left(1 - \sigma\left(w_n'^T w_{mid}\right)\right) \right)$$

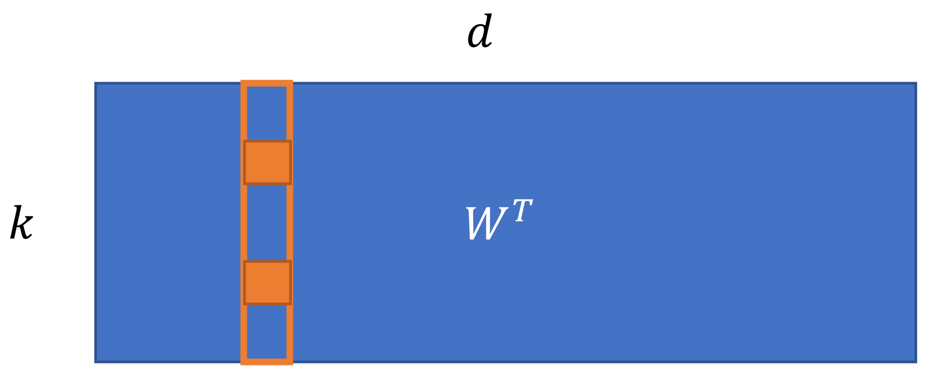
  - where x_i is the one-hot encoding of the i-th word
  - $w_i' = Cx_i$ is the output encoding
  - $w_i = Wx_i$ is the input encoding

LSI and LDA topic models can be seen to produce a representation of words (decomposition approach)

- Let’s examine the topic matrix again



“Automotive” topic may have high values on columns for “car”, “automobile” and “truck”.



Each word can show up in different topics. “book” may show up in “literature” and “school” topics/rows.

More recent word embeddings

- GloVe - https://nlp.stanford.edu/projects/glove/
- BERT - Bidirectional Encoder Representations from Transformers (2018), https://arxiv.org/abs/1810.04805, https://github.com/google-research/bert
- GPT3 (2020), An autoregressive language model with 175 billion parameters https://arxiv.org/abs/2005.14165v2,

Additional resources for word embeddings

- Nice tutorial on word2vec: http://jalammar.github.io/illustrated-word2vec/
- TensorFlow tutorial on Word2vec: https://www.tensorflow.org/tutorials/text/word2vec#next_steps
- Distributional semantics https://en.wikipedia.org/wiki/Distributional_semantics
- Self-supervised learning papers https://github.com/jason718/awesome-self-supervised-learning