

# Aside 5: Word Embedding

# Representing sentences

## ▪ Natural Language

“The man loves his son”

Clearly, this is a *sequence*, of words

*How can word be represented, effectively?*

### First idea: one hot encoding

Given a dictionary of  $W$  words, each word  $w$  could be assigned a unique vector

$$\mathbf{v}_w \in \{0, 1\}^W$$

- *Not particularly efficient: large vectors with almost entirely filled with zeros*
- *The ordering of components will be meaningless: similarities among words will not be represented*

# Representing sentences

## ▪ Natural Language

“The man loves his son”

Clearly, this is a *sequence*, of words

*How can word be represented, effectively?*

**Would be nice to have: *similarity* among words**

*Cosine similarity* between two vectors

$$\frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} \in [-1, 1]$$

- *Similar words (e.g., “son”, “daughter”) should have a small value – high similarity*

# Representing words

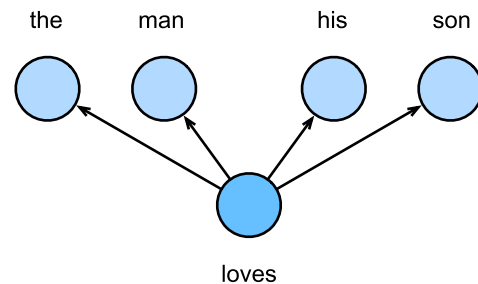
## ■ The Skip-Gram Model

“The man loves his son”

Basic idea: *representing words via their context (in terms of conditional probability)*

$$P(\text{“the”}, \text{“man”}, \text{“his”}, \text{“son”} \mid \text{“loves”})$$

Assuming conditional independence (akin *Naive Bayesian Classifier*):



the following factorization is correct:

$$= P(\text{“the”} \mid \text{“loves”})P(\text{“man”} \mid \text{“loves”})P(\text{“his”} \mid \text{“loves”})P(\text{“son”} \mid \text{“loves”})$$

*Note that the ordering of context words is irrelevant*

# Representing words

## ▪ The Skip-Gram Model

$$P(\text{“the”} \mid \text{“loves”})P(\text{“man”} \mid \text{“loves”})P(\text{“his”} \mid \text{“loves”})P(\text{“son”} \mid \text{“loves”})$$

Conditional probability *factors* are defined via *softmax*

$$P(w_o \mid w_c) := \frac{\exp(\mathbf{u}_o \cdot \mathbf{v}_c)}{\sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c)}$$

under these assumptions:

- each word  $i$  in the dictionary is associated to two vectors  $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^d$
- $\mathbf{v}_i$  is the vector for  $i$  as center word, whereas  $\mathbf{u}_i$  is the vector for  $i$  as context word
- the dimension  $d$  of vectors is an hyperparameter

# Representing words

## ■ The Skip-Gram Model

A **skip-gram** is a *context* of words in a sentence, corresponding to a '*center*' word

Each skip-gram is obtained from a fixed *window size*, that is, the number of words the context of the *center* word

Each skip-gram (a data item) is of the kind (**center\_word, context word**)

## Negative Sampling

A dataset for word embedding can be augmented using *negative sampling*: creating skip-grams for words that *do not* occur with the context of the center word in the sentence

Therefore, a skip-gram becomes

(**center\_word, context word, label**)  
where **label** is either 1 (positive) or 0 (negative)

Window Size	Text	Skip-grams
2	[ The wide road shimmered ] in the hot sun.	wide, the wide, road wide, shimmered
	The [ wide road shimmered in the ] hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [ the hot sun ].	sun, the sun, hot
3	[ The wide road shimmered in ] the hot sun.	wide, the wide, road wide, shimmered wide, in
	[ The wide road shimmered in the hot ] sun.	shimmered, the shimmered, wide shimmered, road shimmered, in shimmered, the shimmered, hot
	The wide road shimmered [ in the hot sun ].	sun, in sun, the sun, hot

# Representing words

## ▪ Skip-gram: loss function

Given the independence conditions, the likelihood of a textual sentence of length  $T$  is:

$$\prod_{t=1}^T \prod_{j \in \text{ctx}(j)} P(w^{(j)} \mid w^{(t)})$$

where  $\text{ctx}(j)$  is the context (of fixed length) of word  $j$

Using log probability:

$$\sum_{t=1}^T \sum_{j \in \text{ctx}(j)} \log P(w^{(j)} \mid w^{(t)})$$

where:

$$\log P(w_o \mid w_c) = \mathbf{u}_o \cdot \mathbf{v}_c - \log \left( \sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c) \right)$$

# Representing words

## ▪ Skip gram: gradient

$$\begin{aligned}\frac{\partial \log P(w_o | w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j=1}^W \exp(\mathbf{u}_j \cdot \mathbf{v}_c) \mathbf{u}_j}{\sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c)} \\ &= \mathbf{u}_o - \sum_{j=1}^W \left( \frac{\exp(\mathbf{u}_j \cdot \mathbf{v}_c)}{\sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c)} \right) \mathbf{u}_j \\ &= \mathbf{u}_o - \sum_{j=1}^W P(w_j | w_c) \mathbf{u}_j\end{aligned}$$



# Representing words

## ▪ Skip-gram: gradient

$$\begin{aligned}\frac{\partial \log P(w_o | w_c)}{\partial \mathbf{u}_o} &= \mathbf{v}_c - \frac{\exp(\mathbf{u}_o \cdot \mathbf{v}_c) \mathbf{v}_c}{\sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c)} \\ &= \mathbf{v}_c - \frac{\exp(\mathbf{u}_j \cdot \mathbf{v}_c)}{\sum_{i=1}^W \exp(\mathbf{u}_i \cdot \mathbf{v}_c)} \mathbf{v}_c \\ &= \mathbf{v}_c - P(w_o | w_c) \mathbf{v}_c\end{aligned}$$

# Representing words

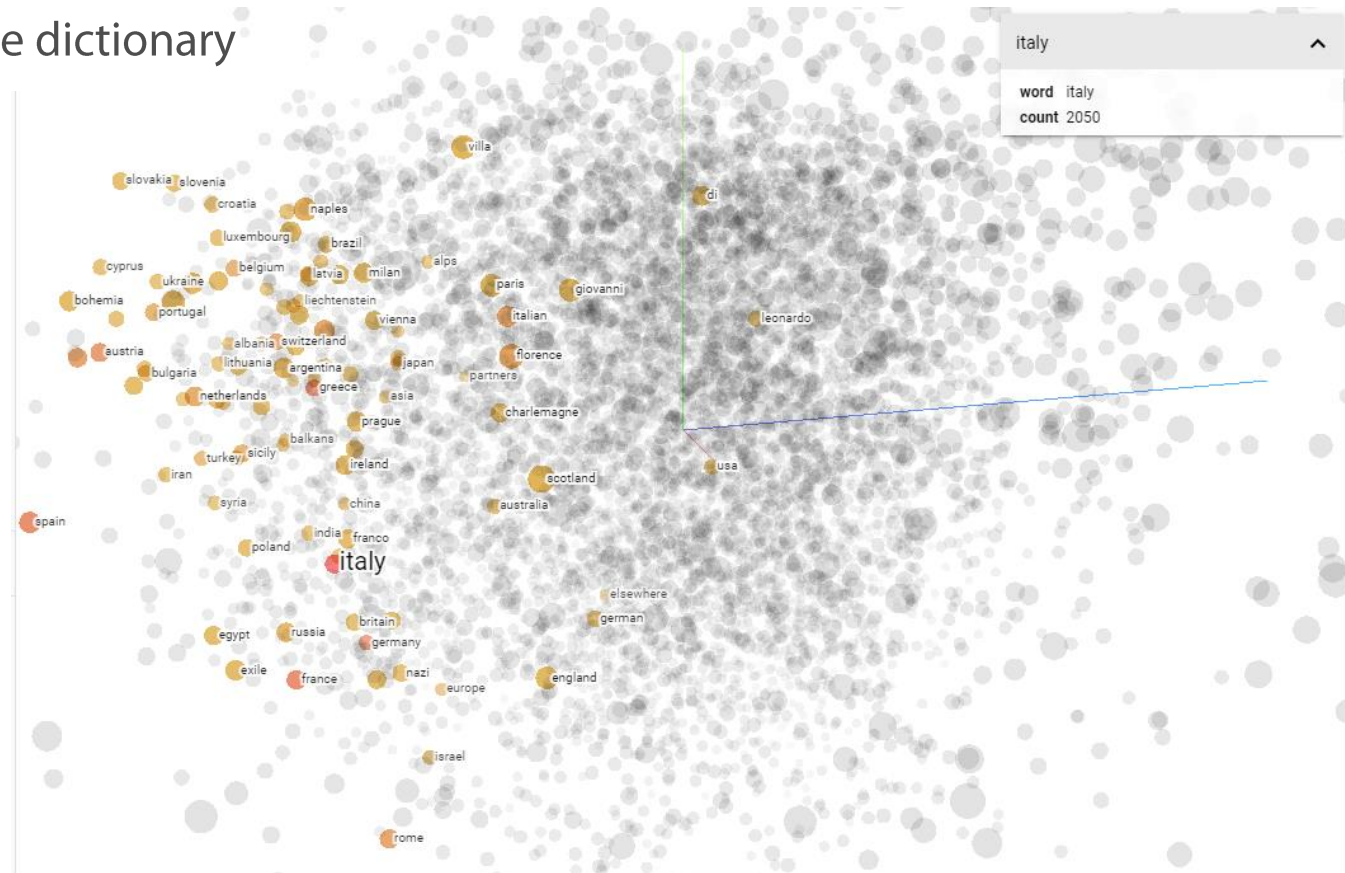
## ■ Training and results

1. Have a dataset (text corpus) of sentences
2. Extract skip-grams, both positive and negative
3. Train with the model with a gradient descent variant
4. Obtain vectors  $\mathbf{v}_i$  and  $\mathbf{u}_i$  for each word in the dictionary
5. Use vectors  $\mathbf{v}_i$  as the *embedded representation* of corresponding words

The dictionary  $\mathcal{W}$  is now represented by vectors whose relative position in a  $d$ -dimensional space reflects the co-occurrence in context

$d$  is an hyperparameter

See <http://projector.tensorflow.org/>



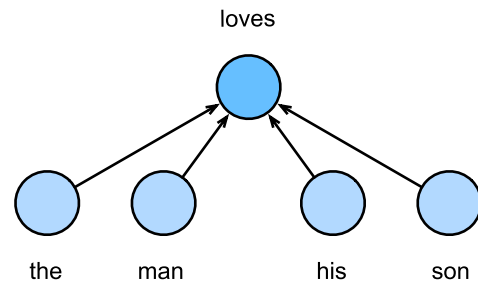
# Representing words

## ■ The Continuous Bag of Words (CBOW) Model

“The man loves his son”

The basic idea is dual to skip-gram: predict center word starting from the context

$$P(\text{“loves”} \mid \text{“the”, “man”, “his”, “son”})$$



Mathematically, this is slightly more complex, since independence assumptions are in the *priors*

Once again, the ordering of context words is irrelevant

# Representing words

## ▪ The Continuous Bag of Words (CBOW) Model

$$P(\text{“loves”} \mid \text{“the”, “man”, “his”, “son”})$$

Conditional probability *factors* are defined via a different *softmax*

$$P(w_c \mid w_{o_1}, \dots, w_{o_m}) = \frac{\exp\left(\frac{1}{m} \mathbf{u}_c \cdot (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_m})\right)}{\sum_{i=1}^W \exp\left(\frac{1}{m} \mathbf{u}_i \cdot (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_m})\right)}$$

From this point on, the derivation is similar.

# Representing words

## ■ word2vec

- Word *vectors* are used to represent words, can also be considered as *feature vectors*
- The technique of mapping words to real vectors is called word *embedding*
- The **word2vec** tool contains both the skip-gram and continuous bag of words models
- The *skip-gram model* assumes that a word can be used to generate its surrounding words in a text sequence
- The *continuous bag of words* model assumes that a center word is generated based on its surrounding context words

## Skip-gram or CBOW?

According to [Mikolov et al., 2013] Skip-Gram works well with small datasets and can better represent less frequent words

However, CBOW is considered to train faster than Skip-Gram and better in representing more frequent words