Aside 6:
Word Embedding

Representing sentences

- **Natural Language**

  "The man loves his son"

  Clearly, this is a *sequence*, of words

  How can each word be represented, effectively?

  First idea: one hot encoding

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

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

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

Representing sentences

- **Natural Language**

“The man loves his son”

Clearly, this is a *sequence*, of words

How can each word be represented, effectively?

Nice-to-have: *similarity* among words

Cosine similarity between two vectors

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

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

Representing words

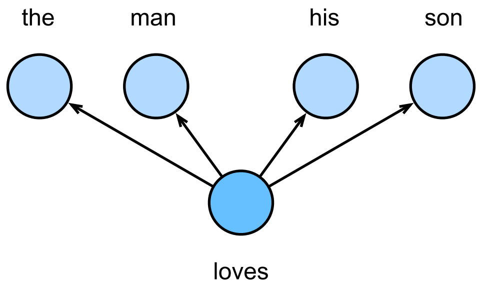
- **The Skip-Gram Model**

"The man loves his son"

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

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

Assuming conditional independence (akin *Naïve 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 <u>irrelevant</u>

[image from http://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html

- **The Skip-Gram Model**

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

Conditional probability *factors* are defined via *softmax*

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

under these assumptions:

- *each word* i *in the dictionary is associated to two vectors* $\boldsymbol{u}_i, \boldsymbol{v}_i \in \mathbb{R}^d$
- $\boldsymbol{v}_i$ *is the vector for* i *as <u>center</u> word, whereas* $\boldsymbol{u}_i$ *is the vector for* i *as <u>context</u> 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 <u>*do not*</u> 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

[images from https://www.tensorflow.org/tutorials/text/word2vec]

- **Skip-gram: loss function**

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

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

where $ctxt(t)$ is the context (of fixed length) of word t

Using log-probability:

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

where:

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

- **Skip gram: gradient**

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

Representing words

- **Skip-gram: gradient**

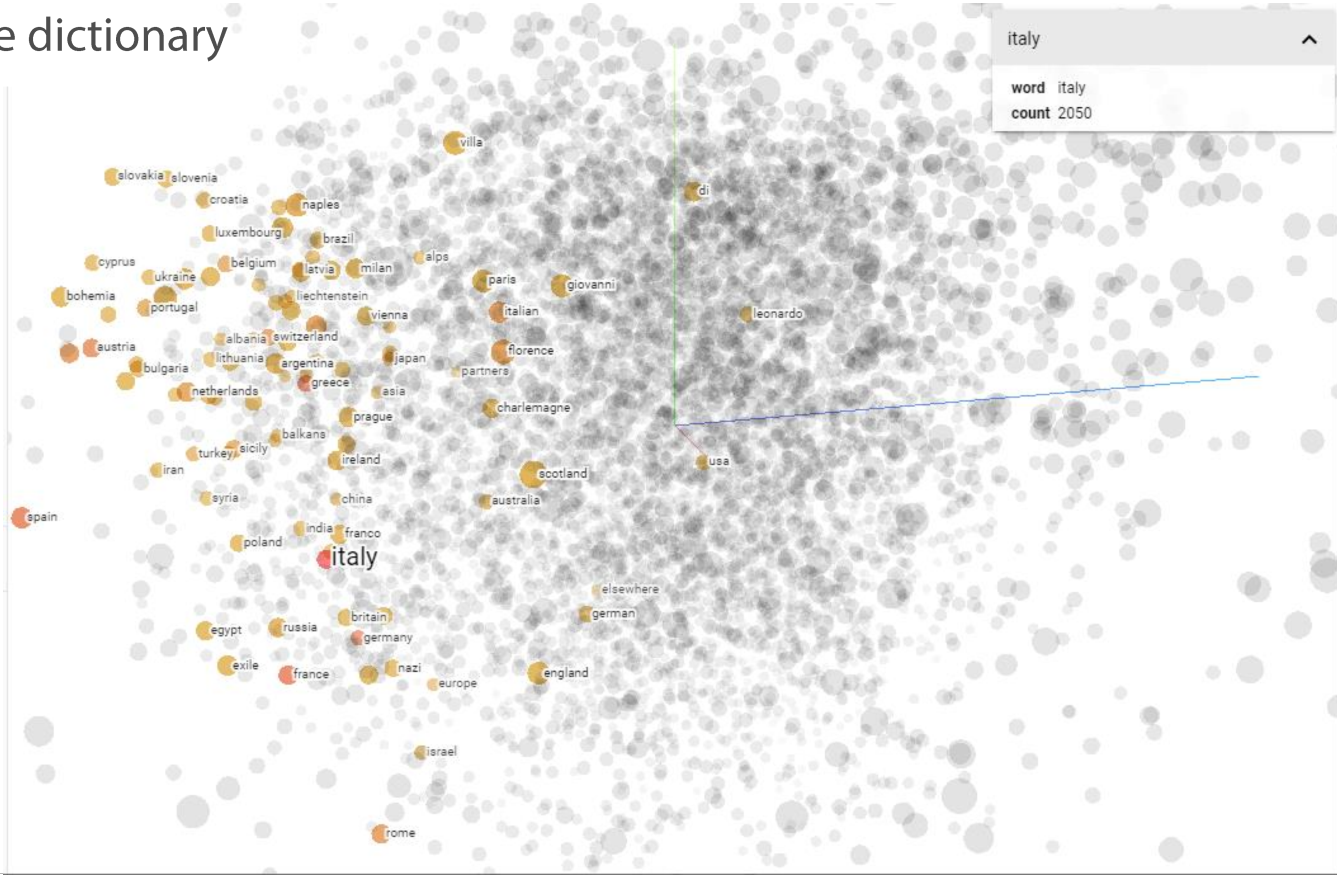
$$
\begin{aligned}
\frac{\partial \log P(w_o \mid w_c)}{\partial \boldsymbol{u}_o} &= \boldsymbol{v}_c - \frac{\exp(\boldsymbol{u}_o \cdot \boldsymbol{v}_c)\boldsymbol{v}_c}{\sum_{i=1}^{W} \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_c)} \\
&= \boldsymbol{v}_c - \frac{\exp(\boldsymbol{u}_j \cdot \boldsymbol{v}_c)}{\sum_{i=1}^{W}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

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 $\boldsymbol{v}_i$ and $\boldsymbol{u}_i$ for each word in the dictionary
  5. Use vectors $\boldsymbol{v}_i$ 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



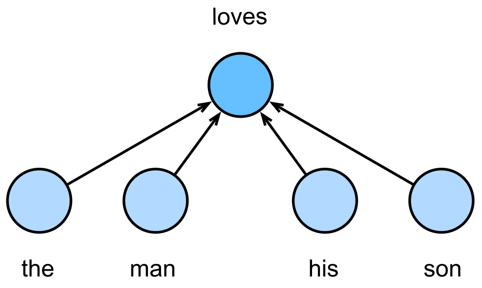
See http://projector.tensorflow.org/

- **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

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



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

Once again, the ordering of context words is <u>irrelevant</u>

[image from http://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html

Representing words

- **The Continuous Bag of Words (CBOW) Model**

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

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

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

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