Dive into Deep Learning for NLP

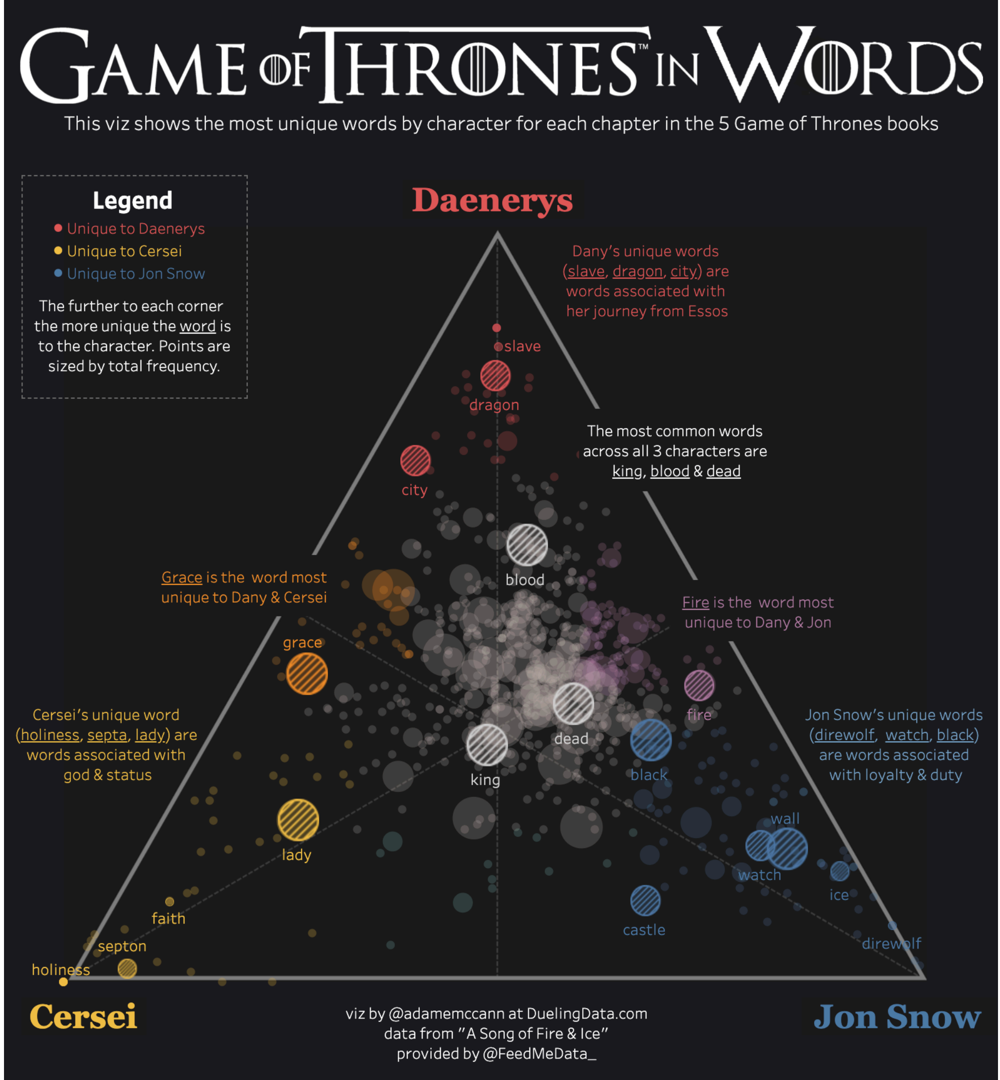
4. Word Embeddings and Applications of Basic Models

Leonard Lausen

gluon-nlp.mxnet.io

13:15-14:15	Natural Language Processing and Deep Learning Basics
14:15-14:25	Break
14:25-15:15	Word Embeddings and Applications of Basic Models
15:15-15:55	Machine Translation and Sequence Generation
15:55-16:35	Contextual Representations with BERT
16:35-16:45	Break
16:45-17:15	Model Deployment with TVM

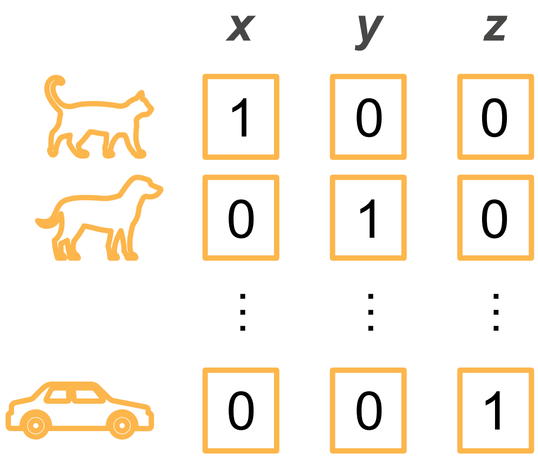
Word Embeddings



Motivation

- One-hot vectors map objects/ words into fixed-length vectors
- These vectors only contain the identity information, not semantic meaning, e.g.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{z}, \mathbf{y} \rangle = 0$$



Word Embeddings

- Learn an embedding vector for each word
- Use $\langle \mathbf{x}, \mathbf{y} \rangle$ to measure the similarity

$$\langle \mathbf{x}, \mathbf{y} \rangle > \langle \mathbf{z}, \mathbf{y} \rangle$$

Word Embeddings

- Learn an embedding vector for each word
- Use $\langle \mathbf{x}, \mathbf{y} \rangle$ to measure the similarity

$$\langle \mathbf{x}, \mathbf{y} \rangle > \langle \mathbf{z}, \mathbf{y} \rangle$$

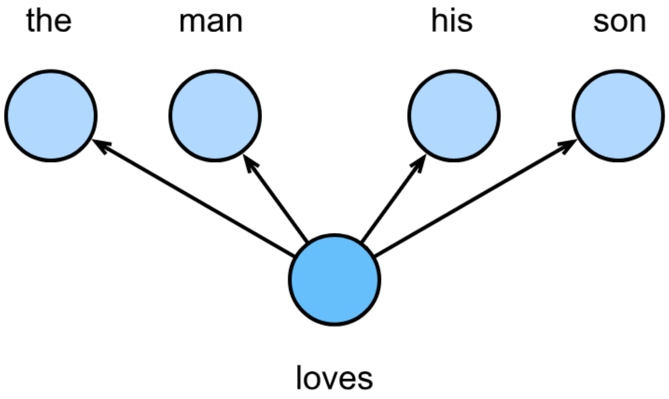
- Requirements on training objective
  - Induces vectors that capture the similarity
  - Cheap to compute
  - Works on unlabeled data

Word2Vec: Skip-Gram

- A word can be used to generate the words surround it
- Given the center word, the context words are generated independently

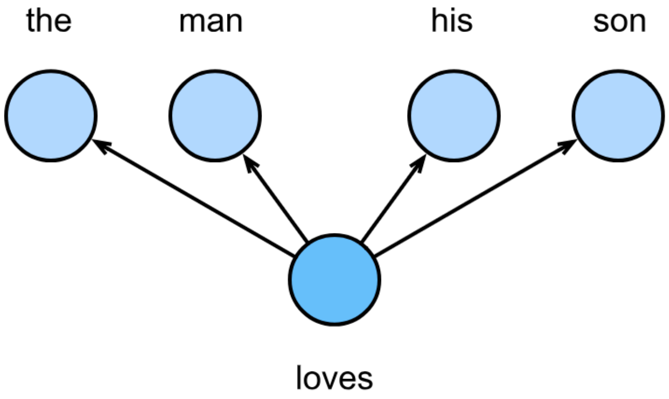
Word2Vec: Skip-Gram

- A word can be used to generate the words surround it
- Given the center word, the context words are generated independently



Word2Vec: Skip-Gram

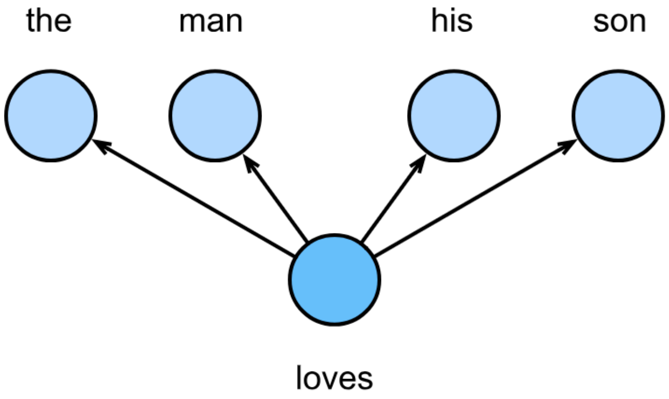
- A word can be used to generate the words surround it
- Given the center word, the context words are generated independently



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

Word2Vec: Skip-Gram

- A word can be used to generate the words surround it
- Given the center word, the context words are generated independently



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

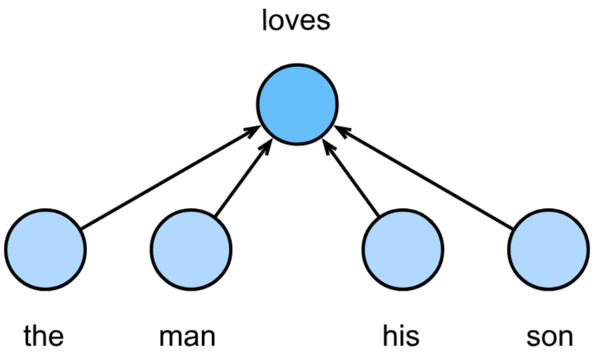
$$= \mathbb{P}(\text{"the"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"man"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"his"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"son"} \mid \text{"loves"})$$

Word2Vec: Continuous Bag Of Words (CBOW)

- The center word is generated based on the context words

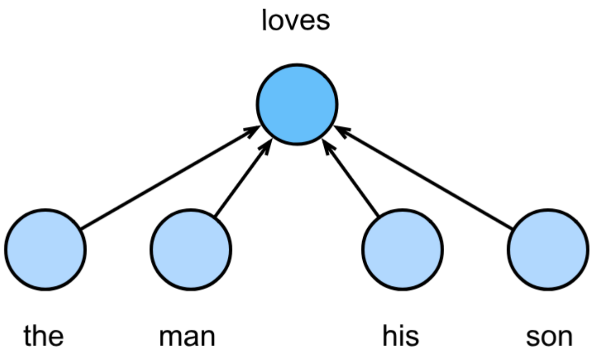
Word2Vec: Continuous Bag Of Words (CBOW)

- The center word is generated based on the context words



Word2Vec: Continuous Bag Of Words (CBOW)

- The center word is generated based on the context words



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

Implementation

	Word	Embedding
Center	w_c	$\mathbf{v}_c \in \mathbb{R}^d$
Context	w_o	$\mathbf{u}_o \in \mathbb{R}^d$

Implementation

	Word	Embedding
Center	w_c	$\mathbf{v}_c \in \mathbb{R}^d$
Context	w_o	$\mathbf{u}_o \in \mathbb{R}^d$

$$\mathbb{P}(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$\mathcal{V}$: all context words

Implementation

Summing over all words is too expensive

	Word	Embedding
Center	w_c	$\mathbf{v}_c \in \mathbb{R}^d$
Context	w_o	$\mathbf{u}_o \in \mathbb{R}^d$

$$\mathbb{P}(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathscr{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$\mathscr{V}$: all context words

Implementation: Negative Sampling

Implementation: Negative Sampling

- Slide window over corpus
- Treat a center word and a context word appear in the same context window as an event
- Sample noise word w_n that doesn't appear in the window

Objective: Binary classification task between noise and true cooccurrences

Word2Vec as Weighted Logistic Matrix Factorization

Word2Vec as Weighted Logistic Matrix Factorization

- Alternative implementation

Word2Vec as Weighted Logistic Matrix Factorization

- Alternative implementation
  - Collect counts in cooccurrence matrix

Word2Vec as Weighted Logistic Matrix Factorization

- Alternative implementation
  - Collect counts in cooccurrence matrix
  - Sample pairs with non-zero counts and noise pairs

Word2Vec as Weighted Logistic Matrix Factorization

- Alternative implementation
  - Collect counts in cooccurrence matrix
  - Sample pairs with non-zero counts and noise pairs
  - Compute gradient of binary classification task weighted by cooccurrence count

FastText

FastText

- English words usually have internal structures and formation methods
  - dog, dogs, dogcatcher

FastText

- English words usually have internal structures and formation methods
    - dog, dogs, dogcatcher
- Each center word is represented as a set of subwords
    - “where” -> “<where>” -> *n*-gram
    - n=3: “<wh", "whe", "her", "ere", “re>”

FastText

- English words usually have internal structures and formation methods
  - dog, dogs, dogcatcher
- Each center word is represented as a set of subwords
  - “where” -> “<where>” -> *n*-gram
  - n=3: “<wh", "whe", "her", "ere", “re>”
- Useful for long but infrequent words
  - e.g. pneumonoultramicroscopicsilicovolcanoconiosis

FastText

- For word *w*, $\mathcal{G}_w$ is the union of subwords with length from 3 to 6
- The center vector is then

$$\mathbf{u}_w = \frac{1}{|\mathcal{G}_w|} \sum_{g \in \mathcal{G}_w} \mathbf{u}_g$$

- The rest of the model is the same as skip-gram