Word Embedding

2018. 5. 9.

Lee, Gyeongbok

DI-LAB
Data Intelligence Laboratory

Contents

- Traditional word representations
  - bag-of-words
  - word co-occurrence
- Word2Vec Basics
  - CBOW? Skip-gram?
  - Improved versions
- Dealing with cross-lingual
  - embedding alignment
- Applications

Introduction

What is this?

Introduction

- Human recognize  as word `tree`
  - With properties: green, tall, plant, long-living, …
  - Maybe in other languages: 木, 나무, Arbre, Baum, дерево, …
- How about machine?
  - tree = `74 72 65 65`
  - Machine can only process numbers orz
  - Machine cannot directly understand semantics from text orz
- New representation of word is needed!
  - As structure containing numerical value (array/vector/matrix)
  - It would be good if the representation has the semantics ☺

Bag-of-words

- Regard word as discrete symbols
  - Ex: animal=10, house=12, plant=31

- Words can be represented as one-hot vector

animal	[0 0 0 0 0 .. 0 1 0 0 0 … 0 0 0 0 0]
house	[0 0 0 0 0 .. 0 0 0 1 0 … 0 0 0 0 0]
plant	[0 0 0 0 0 .. 0 0 0 0 0 … 0 1 0 0 0]

Vector dimension = Number of words

Bag-of-words

- Problem with bag-of-word representation
  - Example: can we get similarity between `house` and `home`?
    how about `house` and `text`? (assume home=3, text=5)

house	[0 0 0 0 0 .. 0 0 0 1 0 ... 0 0 0 0 0]
home	[0 0 1 0 0 .. 0 0 0 0 0 ... 0 0 0 0 0]
text	[0 0 0 0 1 .. 0 0 0 0 0 ... 0 0 0 0 0]

all vectors are orthogonal to each other!
→ similarity = 0

  - This vector representation does not contain semantic

WordNet

- How can we know the semantic?
  - One available solution: using human resource
- WordNet: contains the list of synonyms/hypernyms

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
for synset in wn.synsets("good"):
    print "(%s)" % synset.pos(),
    print ", ".join([l.name() for l in synset.lemmas()])
```

```
(adj) full, good
(adj) estimable, good, honorable, respectable
(adj) beneficial, good
(adj) good, just, upright
(adj) adept, expert, good, practiced,
proficient, skillful
(adj) dear, good, near
(adj) good, right, ripe
…
(adv) well, good
(adv) thoroughly, soundly, good
(n) good, goodness
(n) commodity, trade good, good
```

synonyms

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

hypernyms

Image from [1]

WordNet

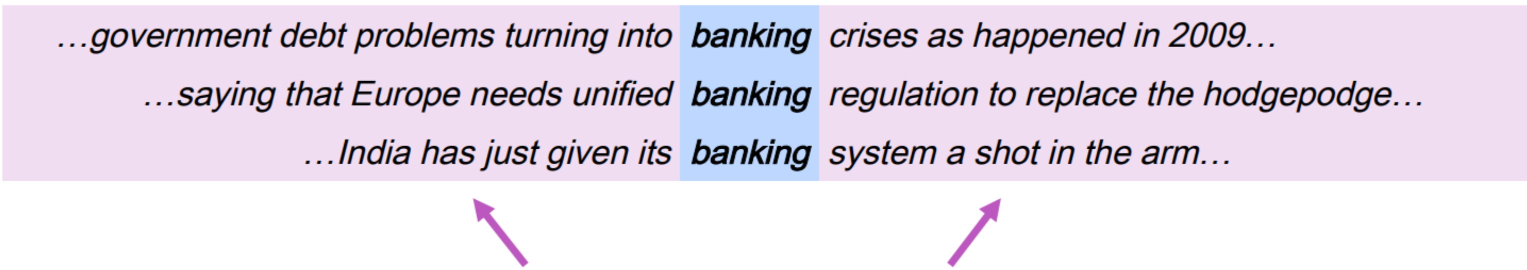
- One way to utilize WordNet?
  - Example: sum of one-hot vector for synonyms/hypernyms?
  - Does the accurate word similarity can be calculated?

- Problems with using human resource
  - Can missing nuance
    - Does `right` is always used as a synonym for `good`?
  - Can missing new meanings of words
    - `wicked` (morally wrong and bad → very, really / excellent)
    - Keeping up-to-date is very hard!
  - Require human labor to maintain (create, adopt)

Words as context

"You shall know a word by the company it keeps"

- Word's meaning is given by words that frequently appear close-by
  - Context: set of words that appears nearby in the fixed-size window (ex: before/after 5 words)
    - Why not full words?



These context words will represent ***banking***

Image from [1]

Words as context

- Full document vs Window
  - Full document: general topics of the word
    - Latent Semantic Analysis
    - Expensive for word representation
  - Window around each word
    - Gives syntactic (Part of Speech), semantic info

- How to make context represent word?
  - Co-occurrence matrix (count-base method)

I like deep learning . / I like NLP . / I enjoy flying .

I - like : 2,
like - deep: 1,
...

Image from [1]

Co-occurrence Matrix Example

I like deep learning . / I like NLP . / I enjoy flying .

window size: 1

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

vector for “like”

Vector dimension = Number of words

Image from [1]

Co-occurrence Matrix Example

I like deep learning . / I like NLP . / I enjoy flying .

window size: 2

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	1	0	1	1	0
like	2	0	0	1	1	1	0	1
enjoy	1	0	0	0	0	0	1	1
deep	1	1	0	0	1	0	0	1
learning	0	1	0	1	0	0	0	1
NLP	1	1	0	0	0	0	0	1
flying	1	0	1	0	0	0	0	1
.	0	1	1	1	1	1	1	0

Vector dimension = Number of words

Problem?

- We can now compare the two word representations

- For long text resource...? @_@

History and relationships to other fields [edit]

into two regions, separated by a linear boundary. Here, it has learned to distinguish black and white circles.

See also: Timeline of machine learning

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM[17]. As a scientific endeavour, machine learning grew out of the quest for artificial intelligence. Already in the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what were then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics.[18] Probabilistic reasoning was also employed, especially in automated medical diagnosis.[19]:488

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation.[19]:488 By 1980, expert systems had come to dominate AI, and statistics was out of favor.[20] Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval.[19]:708–710; 755 Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.[19]:25

Machine learning, reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory.[20] It also benefited from the increasing availability of digitized information, and the ability to distribute it via the Internet.

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on *known* properties learned from the training data, data mining focuses on the discovery of (previously) *unknown* properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples). The difference between the two fields arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.[21]

Relation to statistics [edit]

Machine learning and statistics are closely related fields. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics.[22] He also suggested the term data science as a placeholder to call the overall field.[22]

Leo Breiman distinguished two statistical modelling paradigms: data model and algorithmic model,[23] wherein "algorithmic model" means more or less the machine learning algorithms like Random forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call *statistical learning*.[24]

Theory [edit]

Main article: Computational learning theory

A core objective of a learner is to generalize from its experience.[25][26] Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

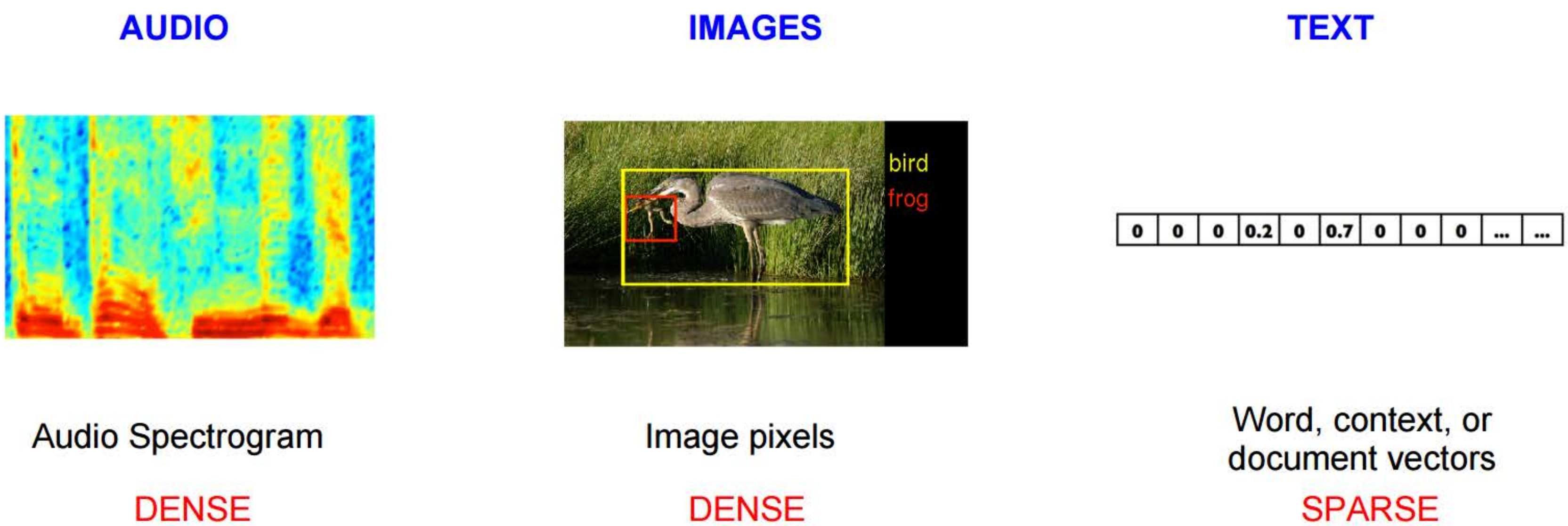
For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.[27]

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

Problem?

- For 10,000 words?
  - 10,000x10,000 matrix (about 4GB)
- The size of the word is much larger in real cases
  - The vector size is too big, but our memory size is limited
  - Sparsity: almost of values are 0
  - Inefficient to process (vector size too big)
- Solution?
  - Sparse matrix representation (CSR, CSC, etc)
  - Dimensionality reduction (PCA, SVD)

Problem?



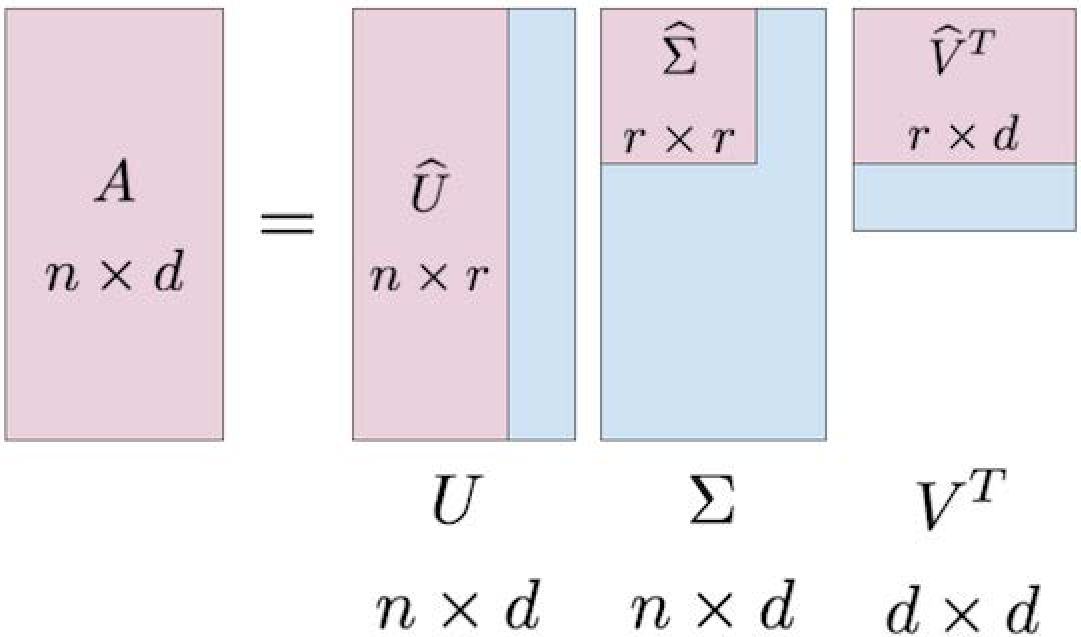
Sparsity: almost of elements are `0`!

(Still has problem of Bag-of-Word representation)

Image from https://www.tensorflow.org/tutorials/word2vec

Dimensionality Reduction

- Store most of the important information in fixed, small number of dimensions (as dense vector)
- Singular Value Decomposition (SVD)



(Problem?)

Image from https://intoli.com/blog/pca-and-svd/

Word Vectors

- Also called word embedding
- Words are now represented as dense vector
  - Expected: words have similar vector representation
    if they has similar context

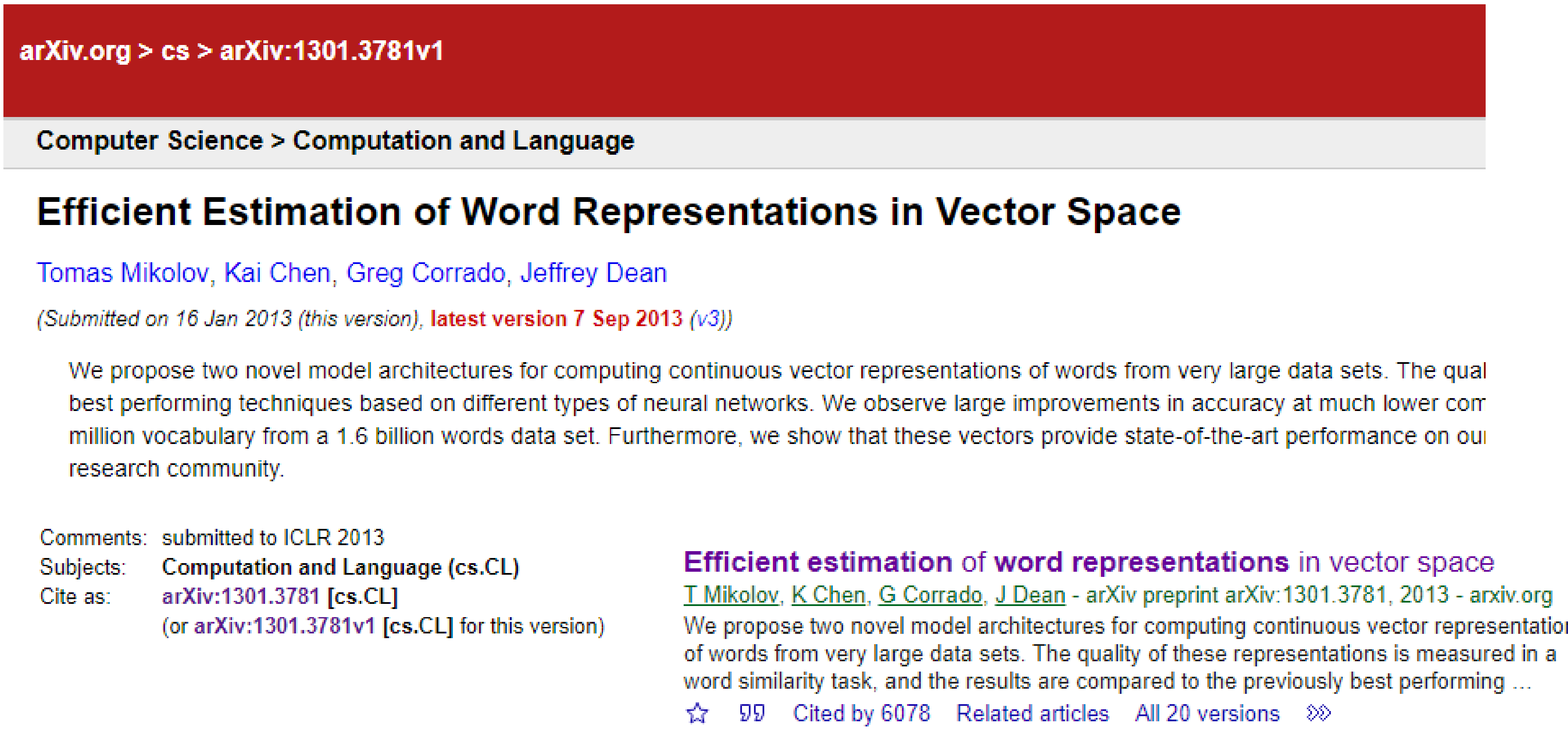
house [0 0 0 0 0 .. 0 0 0 1 0 ... 0 0 0 0 0]
home [0 0 1 0 0 .. 0 0 0 0 0 ... 0 0 0 0 0]
text [0 0 0 0 1 .. 0 0 0 0 0 ... 0 0 0 0 0]

One-hot Vector

house [0.537, 0.596, 0.813, ... , 0.631, 0.681]
home [0.611, 0.237, 0.506, ... , 0.678, 0.672]
text [0.091, 0.322, 0.397, ... , 0.516, 0.283]

Dense Vector

Singularity in 2013



Word2Vec Overview

- Basic: learning word vector from large corpus
  - Every word in fixed vocabulary is represented by a vector
  - Go through each position t in the text, consist of center word c and context words o
  - Similarity of word vectors for c and o are used to calculate the probability of o given c (or c given o)
  - Word vectors are continuously adjusted while training

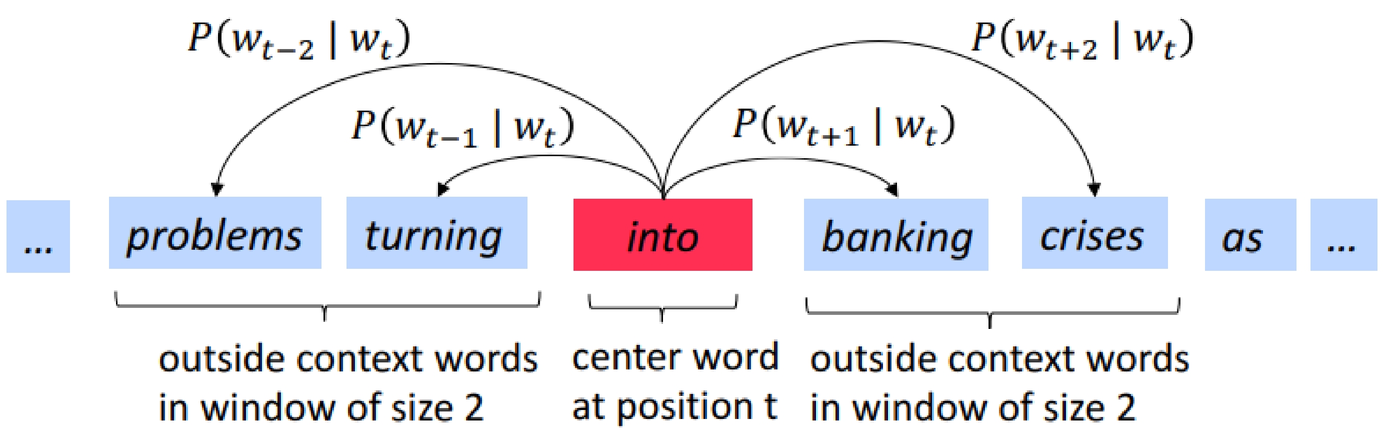


Image from [1]

Word2Vec Overview

- Objective: maximize probability of $P(W_{t+j}|W_t;\theta)$
  - Minimize function $J(\theta)$

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t;\theta\right)$$

  - How to calculate $P(W_{t+j}|W_t;\theta)$?
    - Use two vectors per word: use as center (u) or context (v)

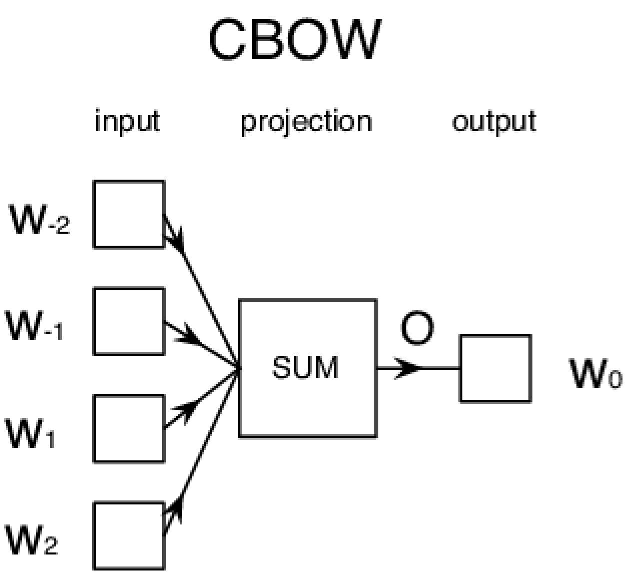
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Dot product compares similarity of *o* and *c*.
Larger dot product = larger probability

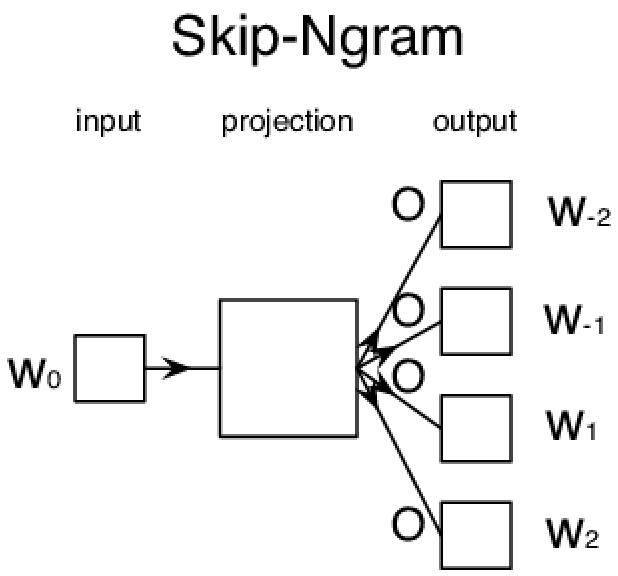
After taking exponent,
normalize over entire vocabulary

CBOW vs Skip-gram

Previous Example

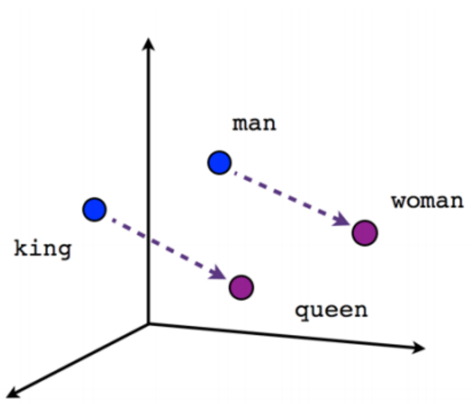


See neighbor words
to predict current word

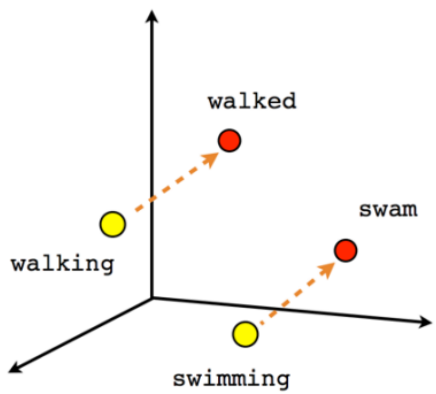


See current word
to predict neighbor words

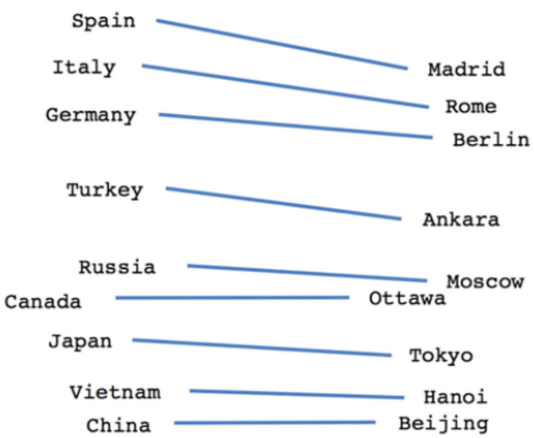
Word2Vec Analogy



Male-Female



Verb tense



Country-Capital

Word2Vec Analogy

Glove results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

Variant of word2vec framework

- Improvements
  - GloVe (utilize global statistics)
- Optimized for specific domain/task
  - Domain: tweet2vec, search2vec, item2vec, etc
  - Task: entity disambiguation, sentiment analysis, etc
- Extend to larger structures
  - Sentence and Document (ex: doc2vec)
- Utilize sub-word information (instead of word)
  - Character-level (ex: charCNN, charRNN)
  - Subword-level (ex: fastText)
- Exploit language-specific features
  - Chinese: radical(部首) / Korean: letter(자모)

Count-base vs Direct Prediction

- LSA, HAL (Lund & Burgess),
- COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- Skip-gram/CBOW (Mikolov et al)
- NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)

- Scale with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Slide from [1]

GloVe

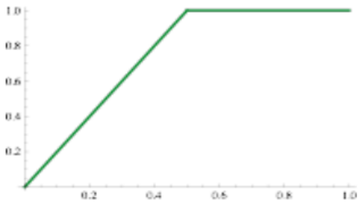
- GloVe: Global Vectors for Word Representation.
  - Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. (Stanford)
  - https://nlp.stanford.edu/projects/glove/

co-occurrence count information

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

$f \sim$



fastText

- Enriching Word Vectors with Subword Information
  - Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, 2016 (Facebook Research)
  - https://arxiv.org/abs/1607.04606

- Using subword information (n-gram)
  - Example: `hungry` → $\{hun, ung, ngr, gry\}$ (with 3-gram)
  - Why subword?
    - Coverage (deal with out-of-vocabulary)
    - Robustness (typo like `hungty`)

Cross-lingual

- We knows:

Tree = 나무

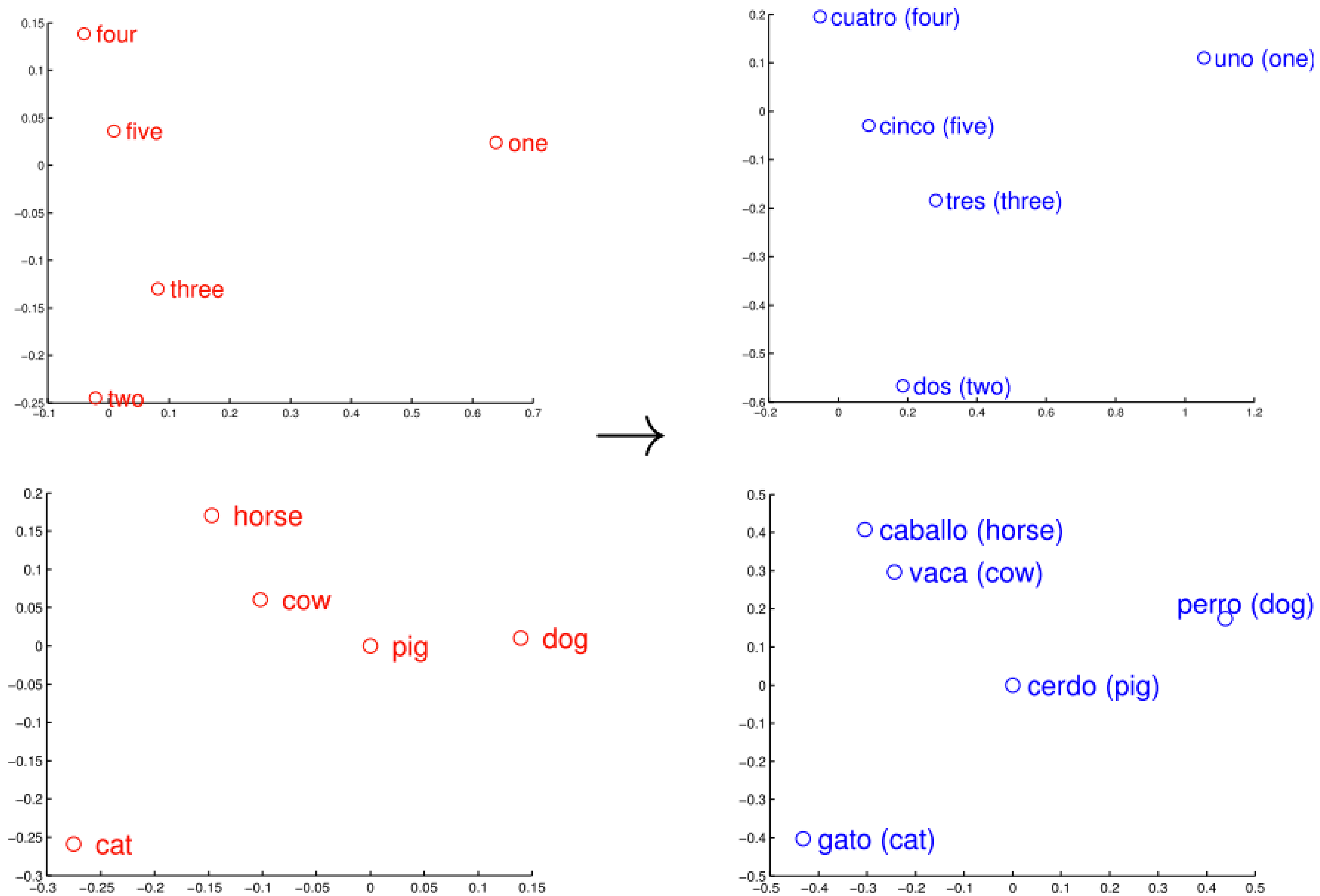
Plant = 식물

- We can make embedding for two language
  - Training individually
  - Can they be directly compared? ex: $\text{sim}(v_{en}(Tree), v_{ko}(\text{식물}))$

- We want to represent these in the same embedding
  - Resource imbalance (English vs other languages)
  - Leverage existing knowledge in English to other language

Relations in Embedding Space

English Spanish



Intuition: similar geometric relations

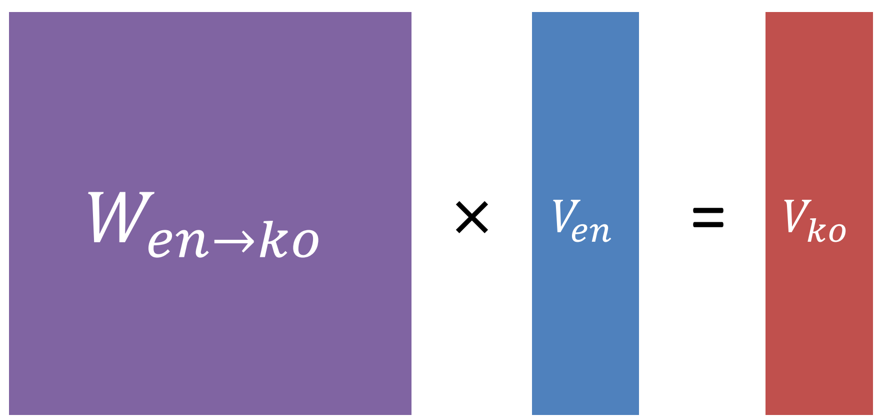
Image from (Mikolov et al., 2013)

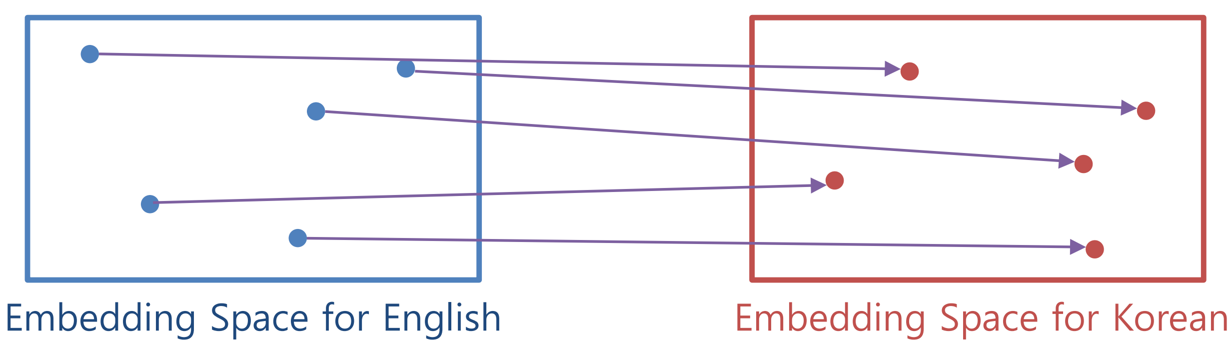
Convert-base method

- Monolingual mapping
  - Train word embedding for each language in large corpus
  - Apply linear transformation after trained with known pairs

Objective:

$$\min_{W} \sum_{i=1}^{n} \|Wx_i - z_i\|^2$$





Various Strategies for cross-lingual

- Make "convert function"
  - Monolingual mapping
- Train with multi-language corpus
  - Pseudo-cross-lingual (mixing contexts)
- Make unique representation
  - Cross-lingual training
  - Optimize cross-lingual constraint to make close each other
- Joint optimization
  - Training the models on parallel, optimize a combination of monolingual & cross-lingual loss

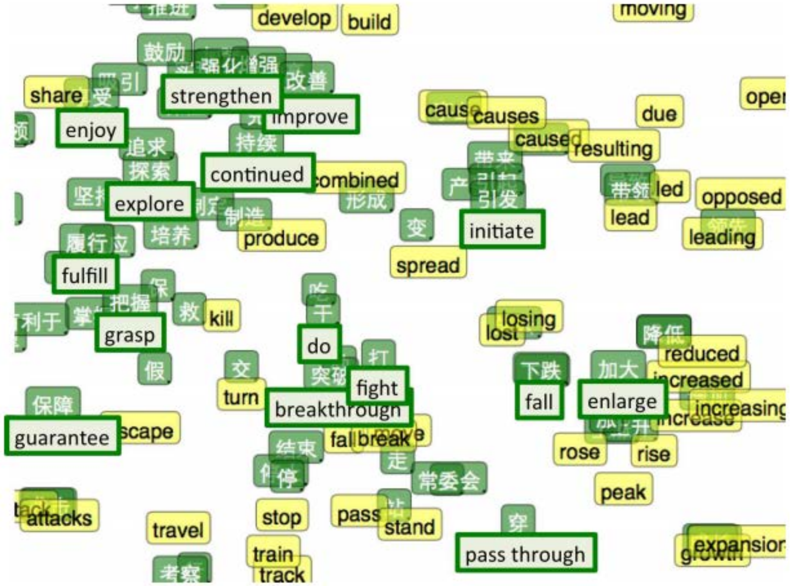
Applications

- Word Similarity
  - Classic method: edit distance, wordnet, stemmer, lemmatization, etc...
    - Stemming (`argue`, `argued`, `argues` → `argu`)
    - Lemmatization (`occurring` → `occur`, `taken` → `take`)
  - Inflections, Tense forms
    - Think, thought, ponder, pondering,
    - Plane, Aircraft, Flight

- With word embedding: get vector similarity (cosine)

Applications

- Machine Translation
  - Classic Methods: Rule-based machine translation, morphological transformation
  - Word embedding can be used as input for NMT
  - Cross-lingual embedding



Applications

- Sentiment Analysis
  - Classifying sentences as positive or negative
  - Classic Methods: Naive Bayes, Random Forests/SVM

  - Building sentiment lexicons using seed sentiment sets
  - We can just use cosine similarity to compare unseen reviews to know reviews. (no classifier!)

  - Can used as a feature for classifier

```
[T to break): sad

abulary: 4067

                   Word       Cosine distance
------------------------------------------------
              saddening              0.727309
                    Sad              0.661083
               saddened              0.660439
           heartbreaking             0.657351
            disheartening            0.650732
           Meny_Friedman             0.648706
 parishioner_Pat_Patello             0.647586
              saddens_me             0.640712
             distressing             0.639909
       reminders_bobbing             0.635772
        Turkoman_Shiites             0.635577
                 saddest             0.634551
            unfortunate              0.627209
                  sorry              0.619405
            bittersweet              0.617521
                 tragic              0.611279
              regretful              0.603472
```

References

- [1] Deep Learning for NLP by Richard Socher
  - http://web.stanford.edu/class/cs224n/
  - Check Syllabus (Slides, etc.) → Word Vectors 1 & 2
- [2] Tutorial and Visualization tool by Xin Rong
  - https://arxiv.org/abs/1411.2738
  - https://ronxin.github.io/wevi/
- [3] A survey of cross-lingual embedding models
  - http://ruder.io/cross-lingual-embeddings/