Natural Language Processing

Edward Choi

Graduate School of AI, KAIST

Index

- Embedding
  - Word/Document embedding
  - Contextualized word embedding
  - Sub-word encoding

Embedding
Word/Document Embedding

What is Word Embedding?

- Express a word as a vector
- 'cat' and 'kitty' are similar words, so they have similar vector representations → short distance
- 'hamburger' is not similar with 'cat' or 'kitty’, so they have different vector representations → far distance

Encoding Categorical Variable

- One-hot encoding or dummy encoding
- Encoding a categorical variable

e.g., a variable with one of the blood types (A, B, AB, and O)

- A = $[\ 1\ 0\ 0\ 0\ ]^T$
- B = $[\ 0\ 1\ 0\ 0\ ]^T$
- AB = $[\ 0\ 0\ 1\ 0\ ]^T$
- O = $[\ 0\ 0\ 0\ 1]^T$.

Pre-existing word representation method

- Each word can be represented by a one-hot encoding which each word takes up its respective dimension.
- horse = [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]^T
- zebra = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]^T

- Inner-product similarity between two different words (e.g., horse and zebra) is a lways 0.
- Euclidean distance between them is always $\sqrt{2}$

- However, ‘horse’ and ‘zebra’ should be semantically similar than ‘horse’ and ‘desk’, since they are living creatures and mammals.

Vectorization for document with pre-existing method

Building a term-document matrix by the method described in previous slides

This document vector is called a bag-of-words vector.

- e.g., Document 1 = “John likes movies. Mary likes too.”

  Document 2 = “John also likes football.”

Vocabulary	Doc 1	Doc 2
John	1	1
likes	2	1
movies	1	0
also	0	1
football	0	1
Mary	1	0
too	1	0

Similarity and distance between the two documents can be obtained by their inner product, cosine similarity, and Euclidean distance.

Other vector representation methods

- Matrix factorization
  - Singular value decomposition (or latent semantic indexing)
  - Nonnegative matrix factorization

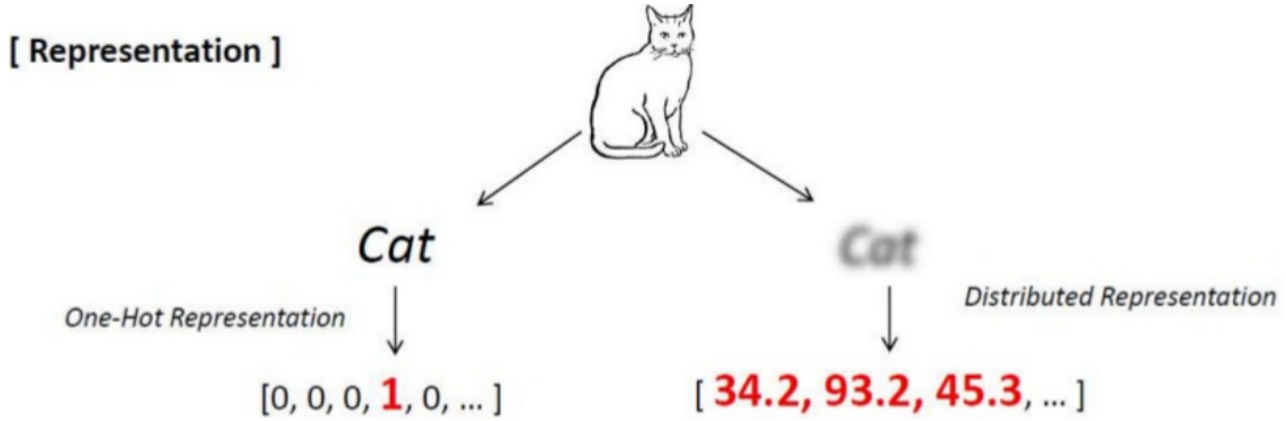
- Probabilistic topic modeling
  - Probabilistic latent semantic indexing
  - Latent Dirichlet allocation

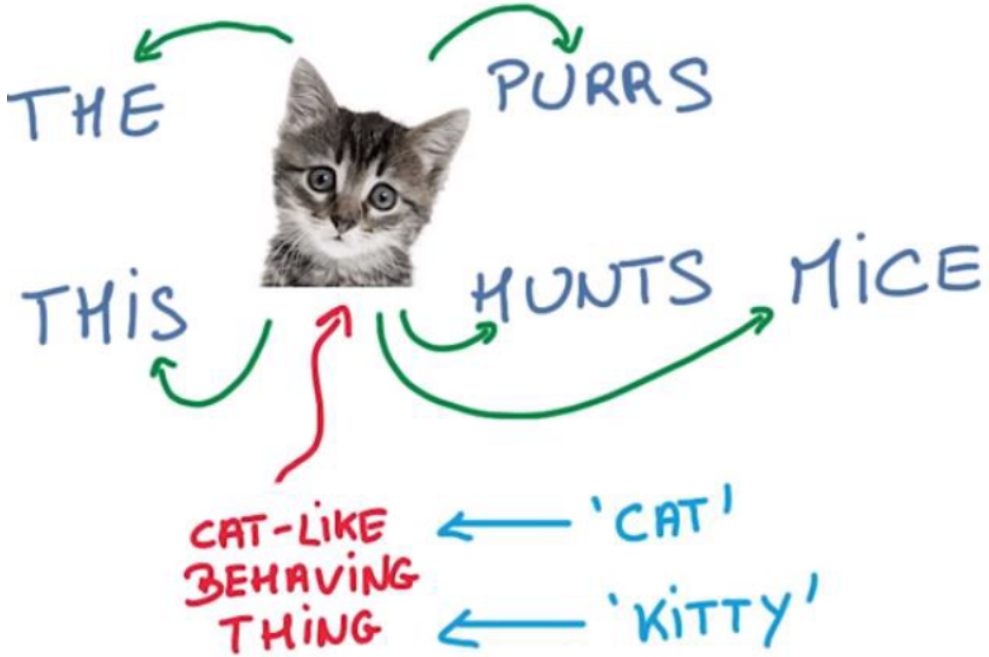
Word Embedding methods

- Distributed vector representations
  - Vector representation in the form of nonzero values across multiple dimensions, as opposed to a conventional one-hot vector.
  - Euclidean distance, inner product, and cosine similarity of two different word vectors encode their semantic similarity
- Topics
  - Word2Vec
  - GloVe
  - Doc2Vec (adaptation of Word2Vec for document vector)

Word2Vec

- An algorithm for training vector representation of a word from context words (adjacent words)
  - Assumption: words in similar context will have similar meanings.
- e.g.)
  - The **cat** purrs.
  - The **cat** hunts mice.



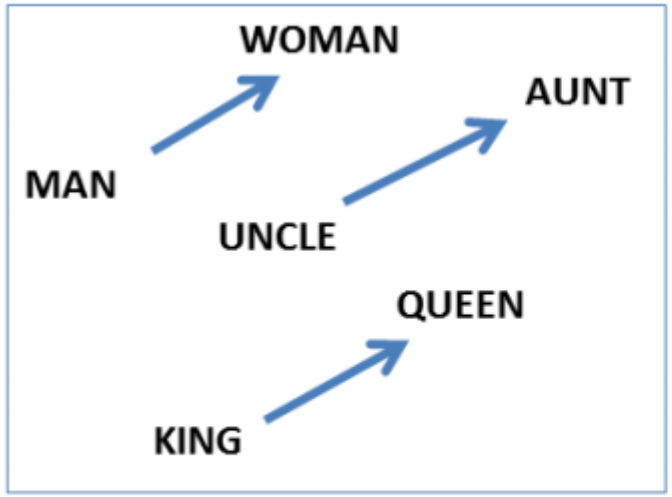


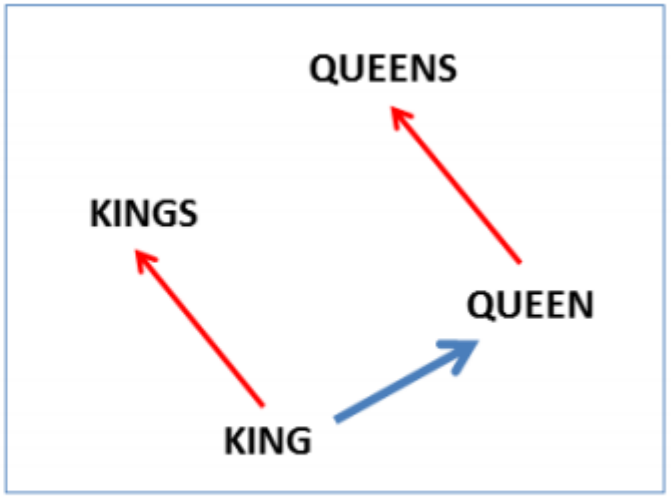
History of Word2Vec

- Learning representations by back-propagating errors (Rumelhart et al., 1986)
- A neural probabilistic language model (Bengio et al., 2003)
- NLP from Scratch (Collobert & Weston, 2008)
- Word2Vec (Mikolov et al., 2013):
  - Efficient Estimation of Word Representation in a Vector Space
  - Distributed Representations of words and phrases and their compositionality
  - Reduced computational/complexity compared to conventional word embedding models based on artificial neural networks (negative sampling, hierarchical softmax)

Property of Word2Vec

- The word vector, or the relationship between vector points in space, represents the relationship between the words.
- The same relationship is represented as the same vectors.





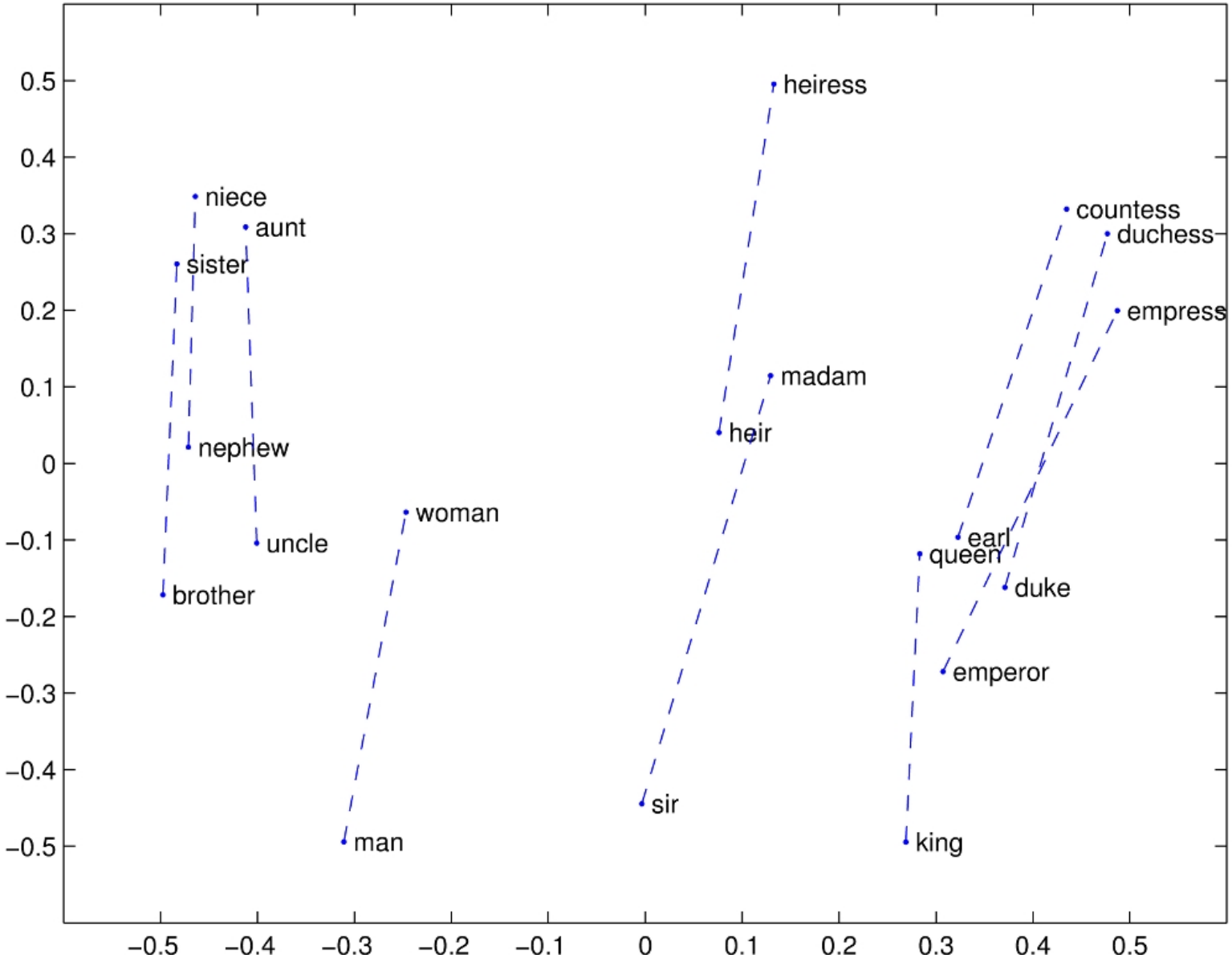
(Mikolov et al., NAACL HLT, 2013)

- e.g.,

vec[queen] – vec[king] = vec[woman] – vec[man]

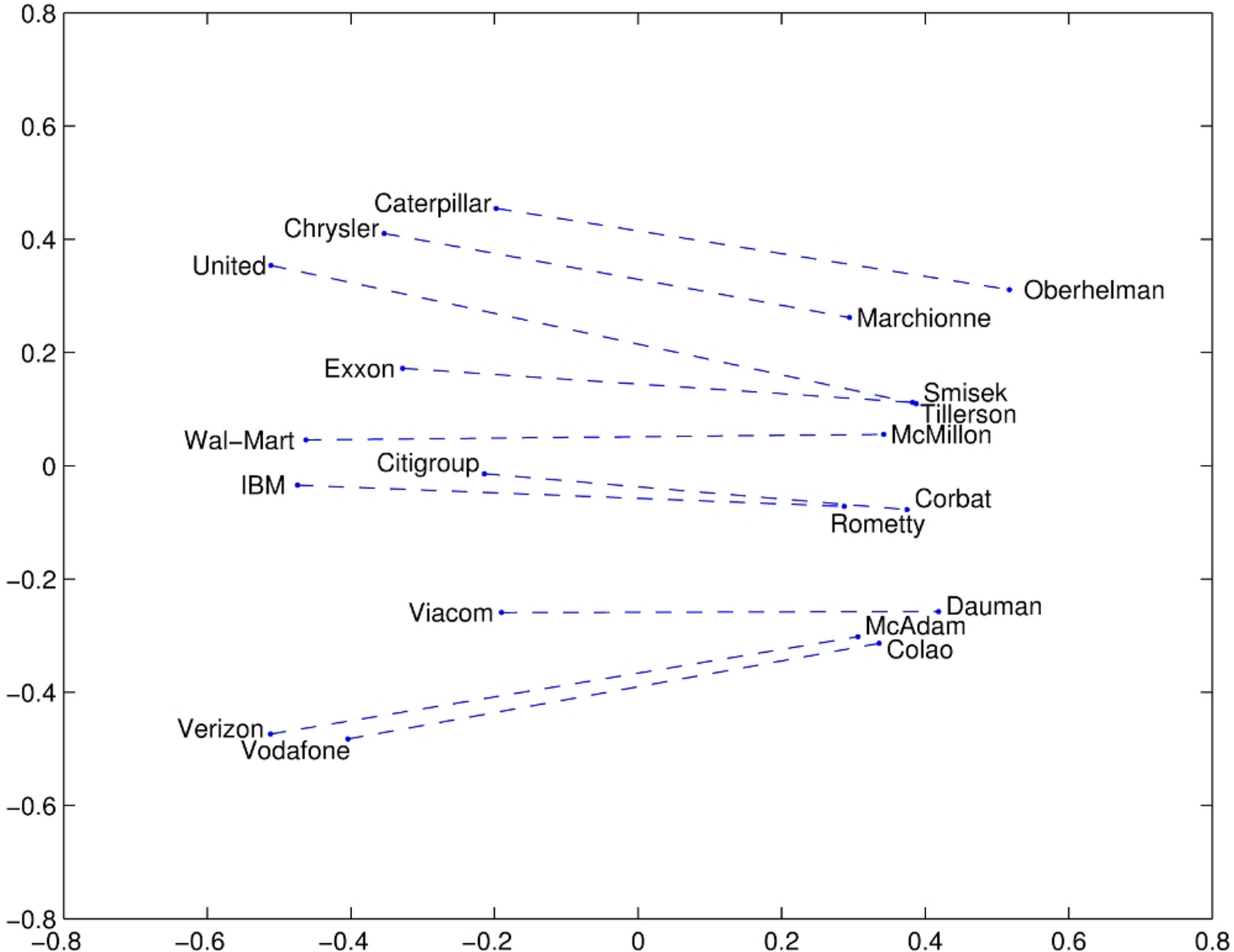
Property of Word2Vec - Linear Substructure

man - woman



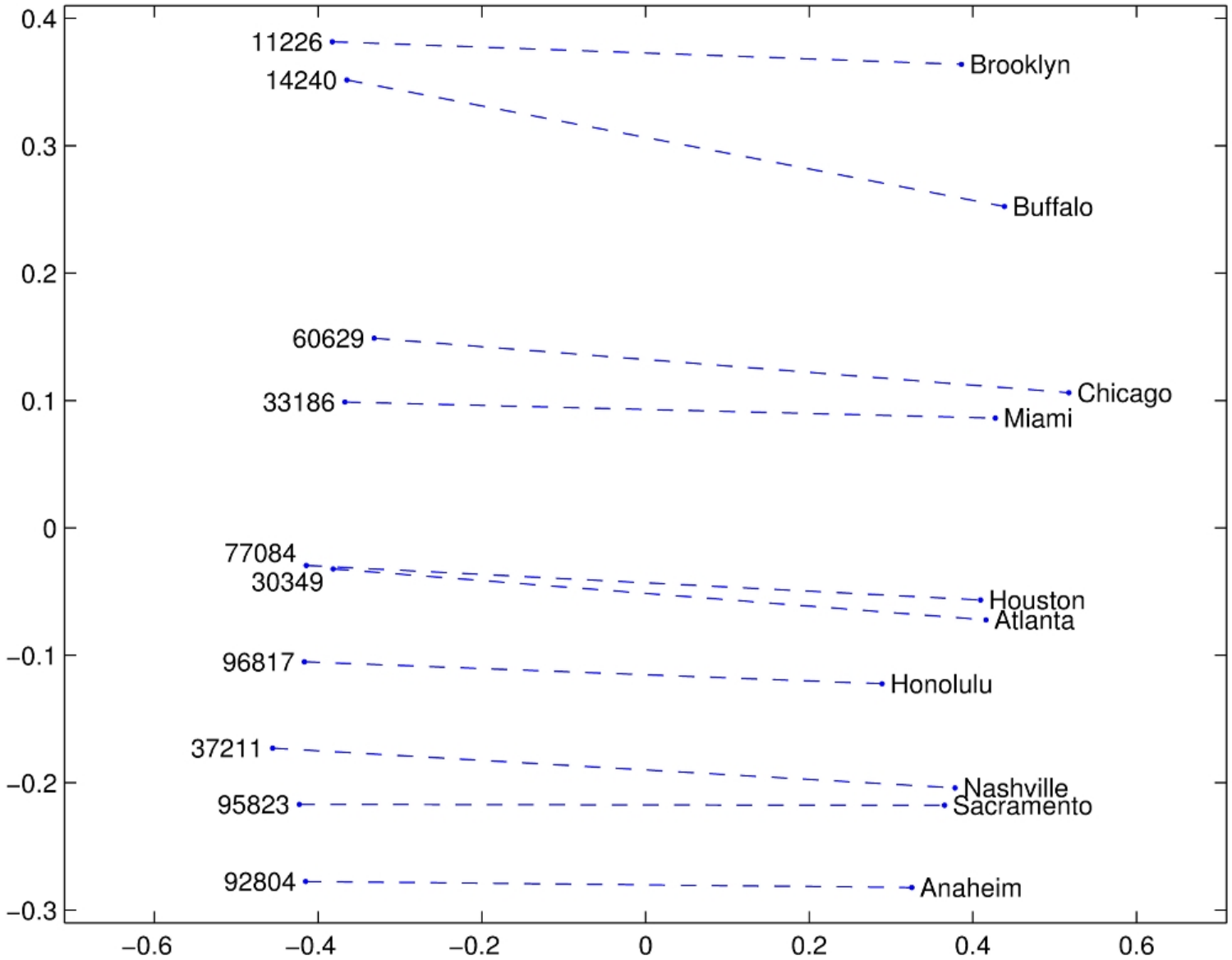
Property of Word2Vec - Linear Substructure

company - ceo



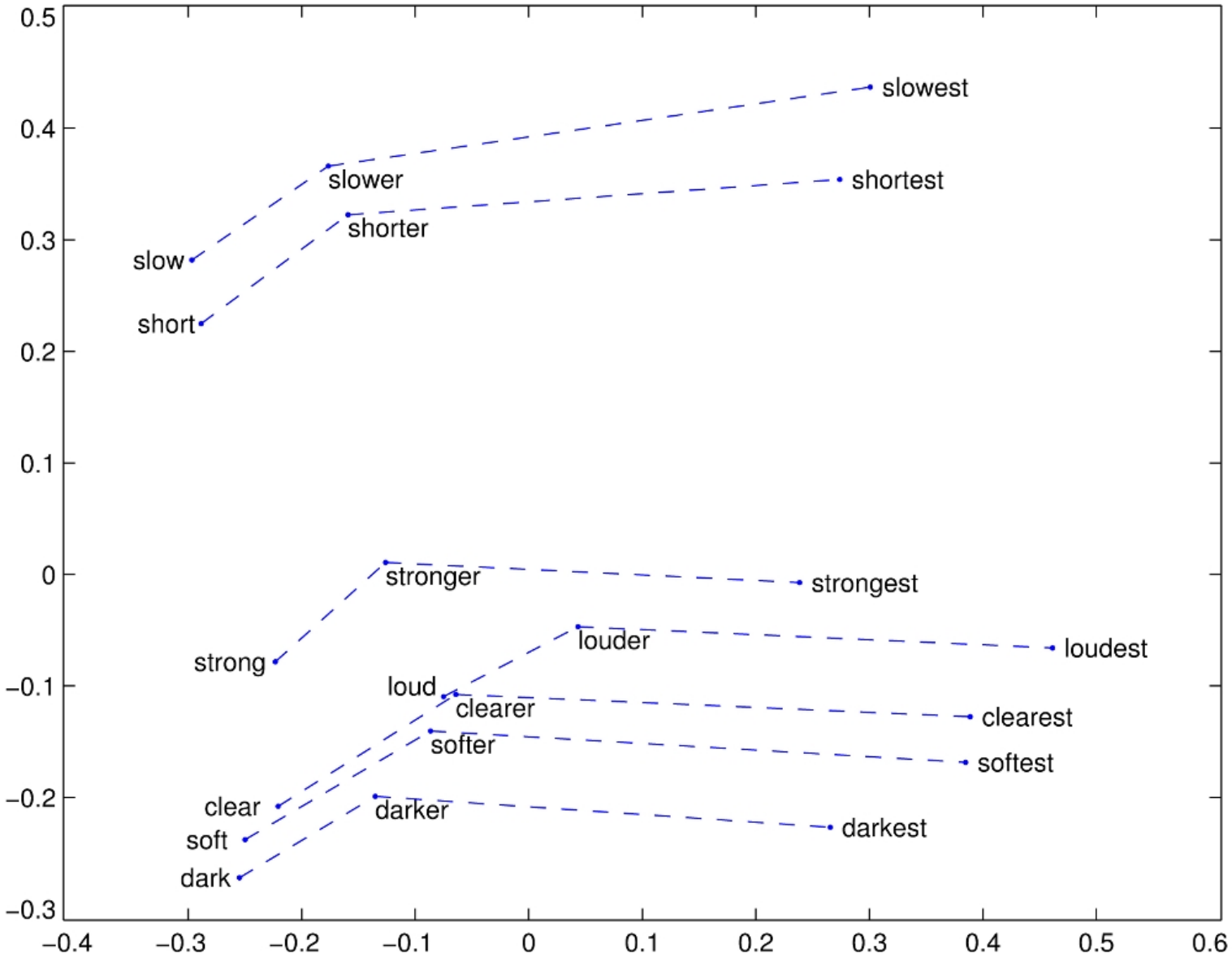
Property of Word2Vec - Linear Substructure

city - zip code



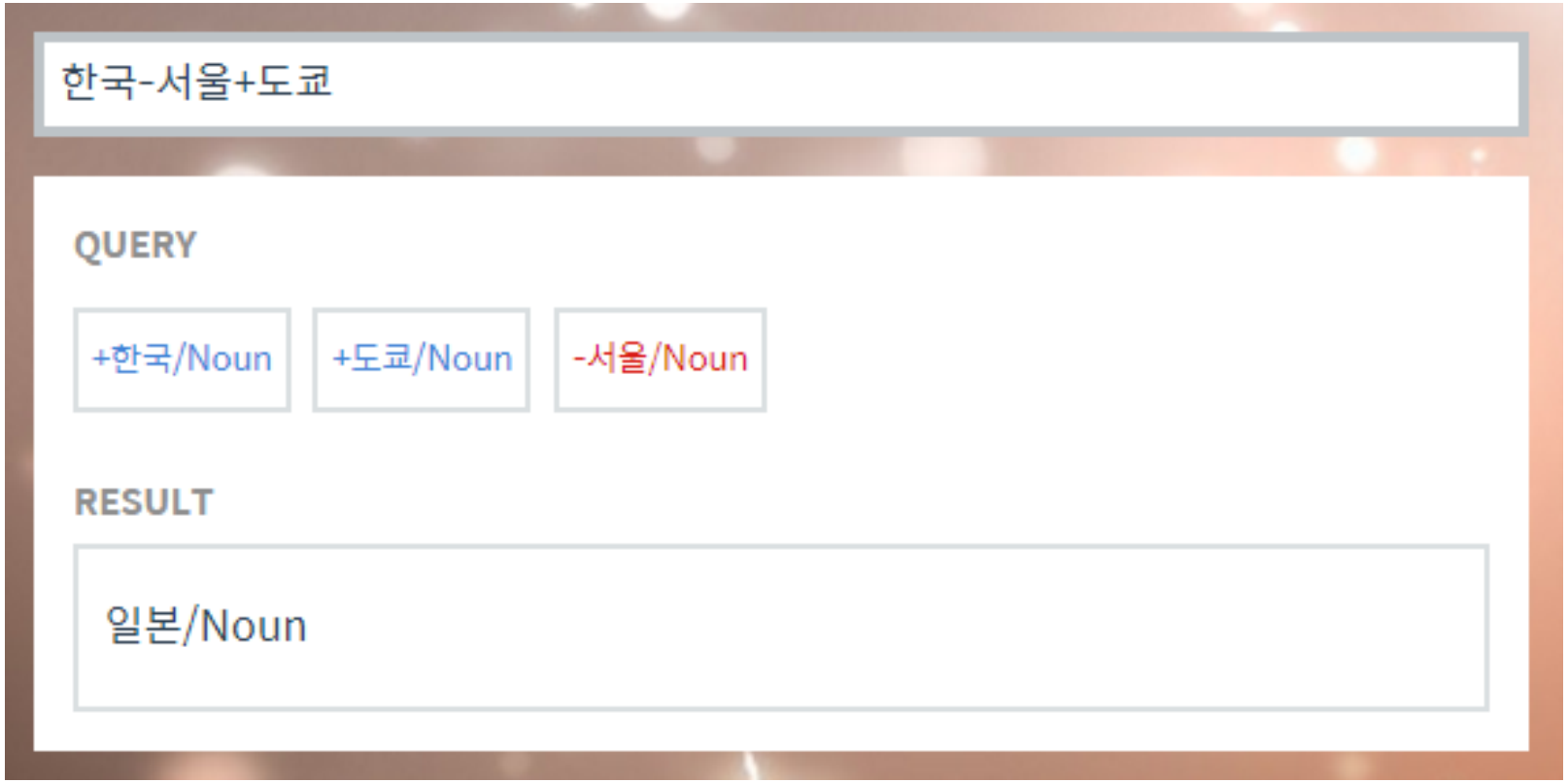
Property of Word2Vec - Linear Substructure

comparative
- superlative



Property of Word2Vec – Analogy Reasoning

- Korean Word2Vec : http://w.elnn.kr/search/



Property of Word2Vec – Analogy Reasoning

- More examples: http://wonjaekim.com/archives/50

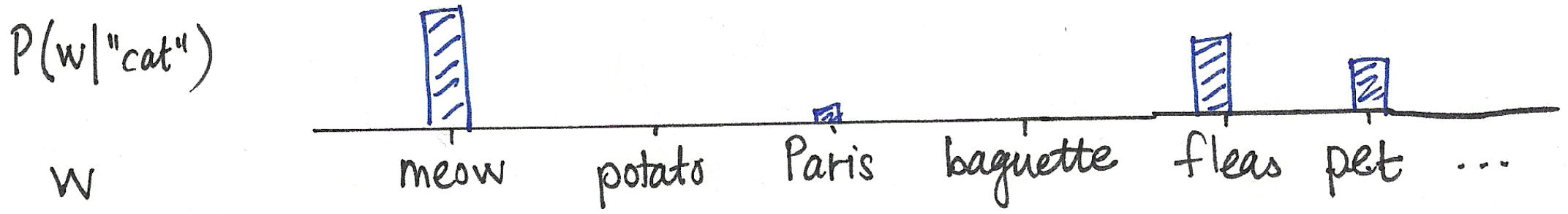
	데모	http://w.elnn.kr/
버락_오바마-미국+러시아	블라디미르/Noun_푸틴/Noun	-
버락_오바마-미국+스타워즈	아나킨/Noun_스카이워커/Noun	-
아카라카-연세대학교+고려대학교	입실렌티/Noun	입실렌티/Noun
아이폰-휴대폰+노트북	아이패드/Noun	아이패드/Noun
컴퓨터공학-자연과학+인문학	법학/Noun	게임학/Noun
플레이스테이션-소니+마이크로소프트	엑스박스/Noun_360/Number	MSX/Alpha
한국-서울+파리	프랑스/Noun	프랑스/Noun
컴퓨터-기계+인간	운영체제/Noun	일반인/Noun
게임+공부	프로그래밍/Noun	덕질/Noun
박보영-배우+가수	애프터스쿨/Noun	허각/Noun
밥+했는지	끓였/Verb	저녁밥/Noun
사랑+이별	그리움/Noun	추억/Noun
삼성-한화	노트북/Noun	후지필름/Noun
소녀시대-소녀+아줌마	아이유/Noun	에이핑크/Noun
수학-증명	경영학/Noun	이산수학/Noun
스파게티-소시지+김치	칼국수/Noun	비빔국수/Noun
아버지-남자+여자	어머니/Noun	어머니/Noun
아이유-노래+연기	송중기/Noun	송중기/Noun
안드로이드-자유	iOS/Alpha	아이폰/Noun
우주-빛	태양계/Noun_밖/Noun	NASA/Alpha
인간-직업	짐승/Noun	볼뉴르크/Noun
최현석_셰프-허세+셰프	이연/Noun_복/Noun	-
패스트푸드-체인점	영국/Noun_요리/Noun	철물/Noun

Idea of Word2Vec

- “You shall know a word by the company it keeps”

  - J. R. Firth 1957

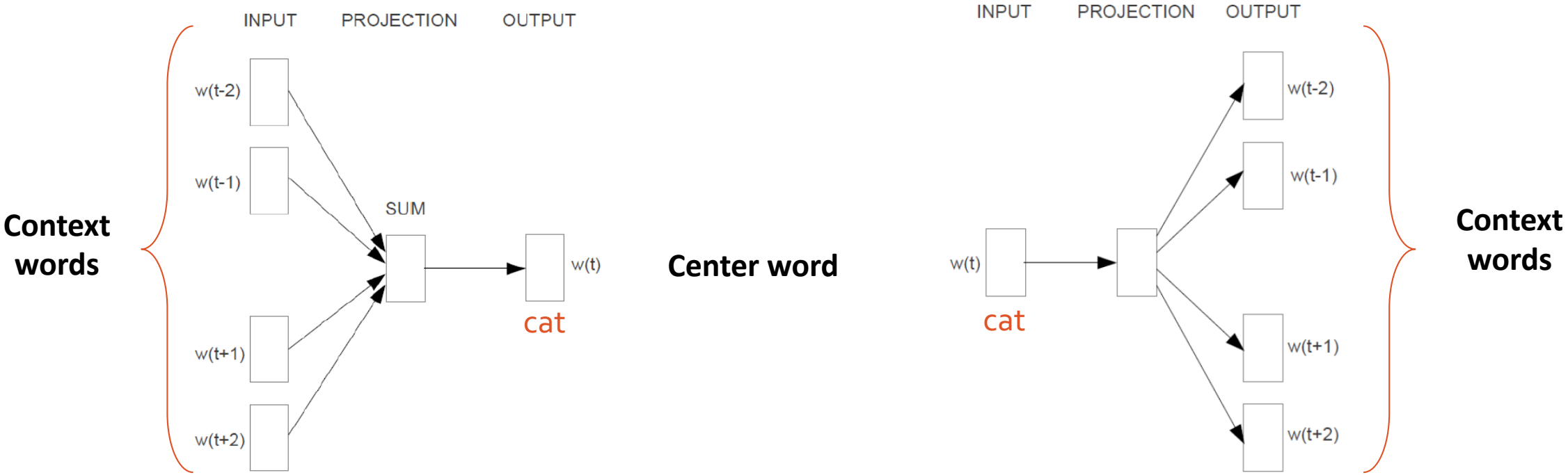
- Suppose we read the word “cat”. What is the probability P(w|cat) that we’ll read the word w nearby?



- Distributional Hypothesis: The meaning of “cat” is captured by the probability distribution P(w|cat).

Two Models of Word2Vec

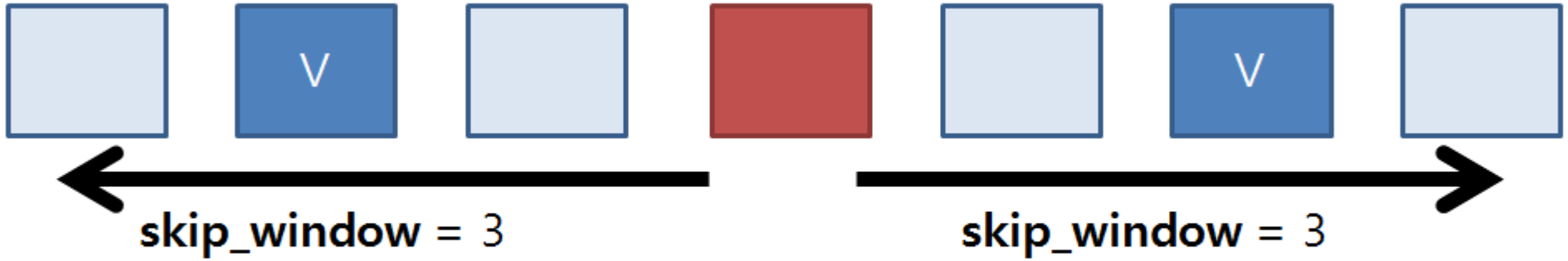
Continuous Bag-Of-Words (CBOW)



Skip-gram

How Word2Vec Algorithm Works

- Select window size first

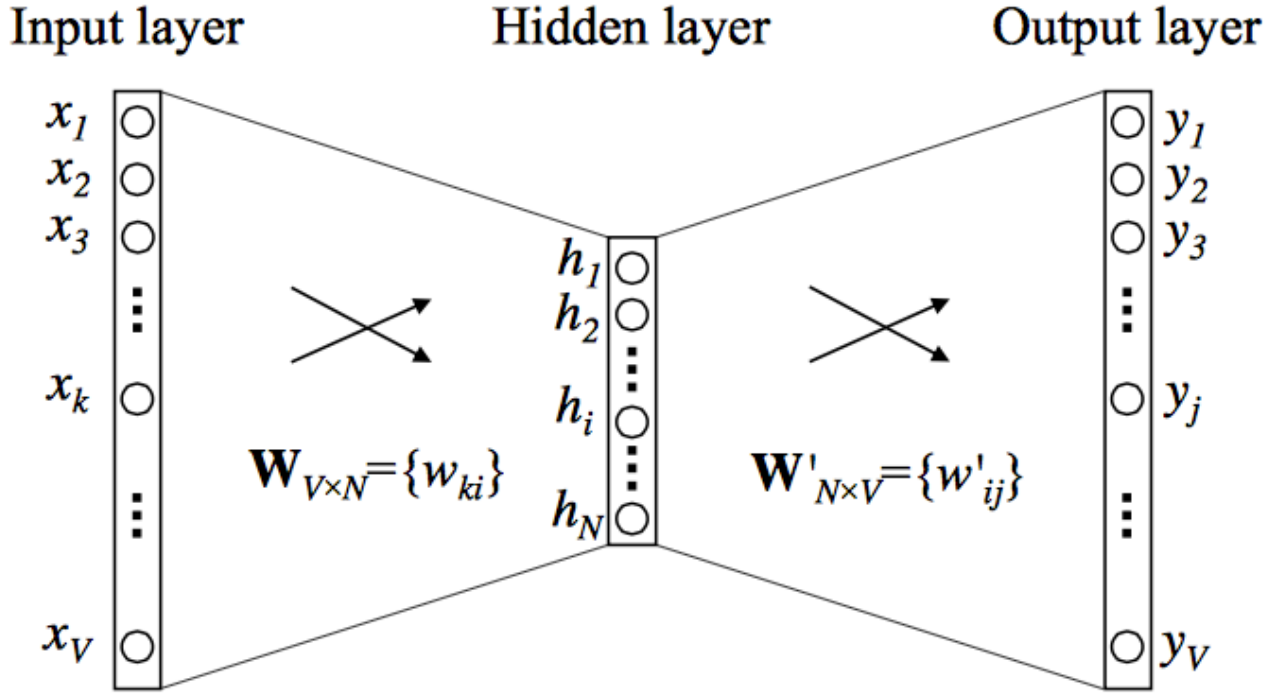


num_skips: #selected words in windows = 2

- Skipgram: predict context words from a center word
  - Good at semantic tasks, but slow in training
- CBOW: predict a center word from context words
  - Good at syntactic tasks, and relatively fast to train

How Word2Vec Algorithm Works

- V : vocabulary size
- N : embedding dimension



- Objective criterion (to maximize)

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

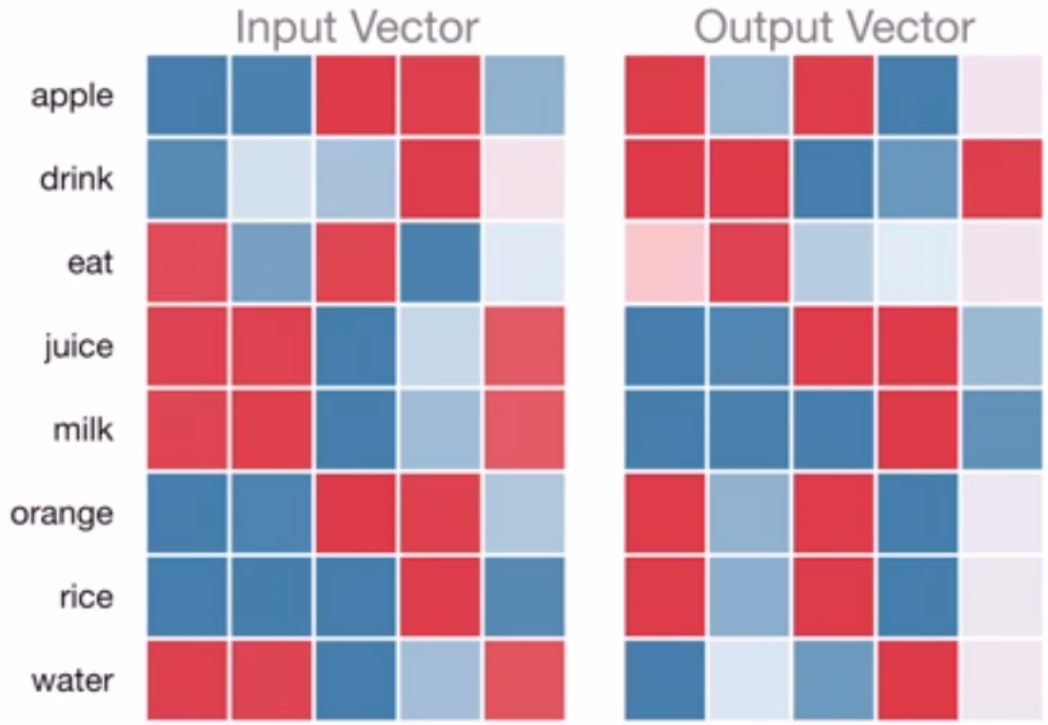
where

$$\log p(o|c) = \log \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$, which is a softmax function.

How Word2Vec Algorithm Works

- A vector representation of ‘eat’ in W_1 has similar pattern with vectors of ‘apple’, ‘orange’, and ‘rice’ in W_2.

- When the input is 'eat', the model can predict 'apple', 'orange’, or 'rice’ for output, because the vectors have high inner product values.

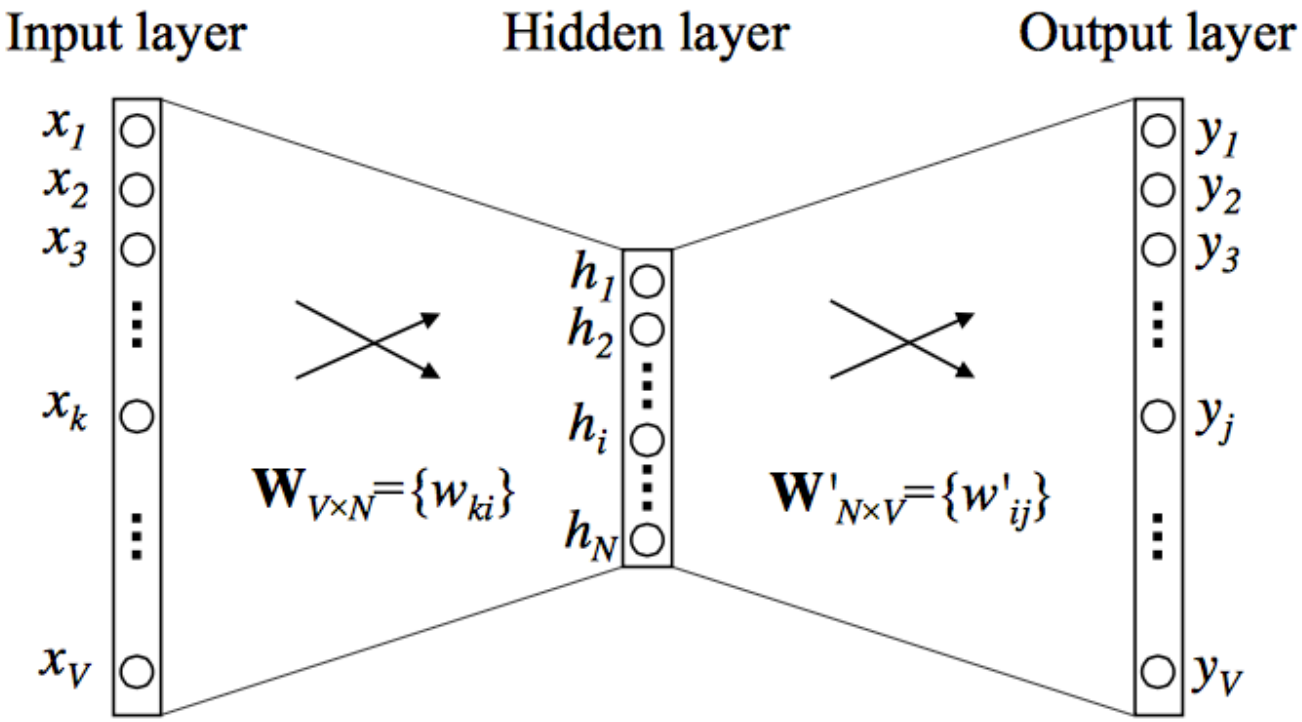


W_1 W_2

https://ronxin.github.io/wevi/

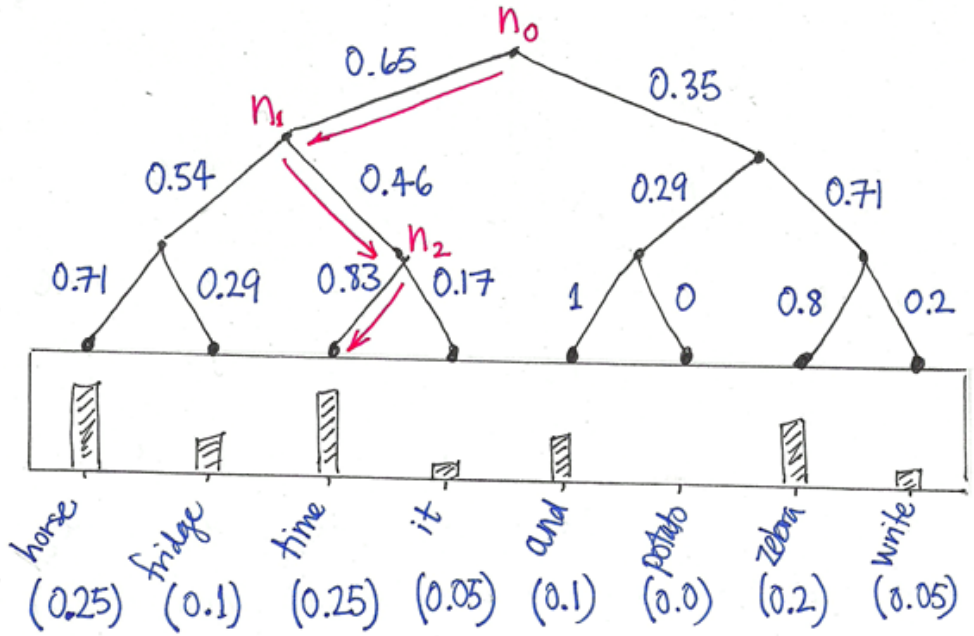
Softmax Implementations are Slow!

- Training parameters W_1 and W_2 is very slow due to a softmax layer.
- The vocabulary size in English is over millions, and all words must be calculated and normalized in order to compute the softmax value in the output layer, so most of the computation resources are used for this increased additional calculation.



Efficient Solution 1: Hierarchical Softmax

- Generate a binary tree of which leaves are each words.
- Calculate the probability of the word by multiplying the probabilities following the path from the root to the leaf.
- i.e., consider each internal node as a Bernoulli random variable that branches to the left and right nodes,and calculate the probability of each word.



$$P(\texttt{time}|\mathcal{C}) = P_{n_0}(\text{left}|\mathcal{C})P_{n_1}(\text{right}|\mathcal{C})P_{n_2}(\text{left}|\mathcal{C})$$

- This reduces the calculation cost from V to $\log V$.

Efficient Solution 2: Negative Sampling

- Alternative method to Hierarchical softmax
- If we randomly sample a small number of words and perform softmax and normalization, the calculation will be reduced from *N* x *V* to *N* x *K* (*K* : number of samples being extracted).
- Words used for groundtruth output are called 'positive samples' because it should be calculated. The key is how to choose the the 'negative samples'.

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

- Negative Sampling defines sampling from 'Noise Distribution' and uses a fixed number of words from its distribution $P_n(w)$, which is experimentally set to 'Unigram Distribution' raised to the ¾th power.

Application of Word2Vec

Word2Vec improves performances in most areas of NLP.

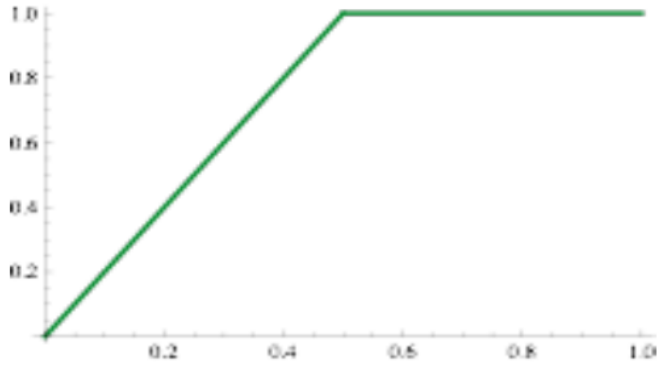
- Word similarity
- Machine translation
- Part-of-speech tagging and named entity recognition
- Sentiment analysis
- Clustering
- Semantic lexicon building

GloVe: Another Word Embedding Model

GloVe: Global Vectors for Word Representation

- Rather than going through each pair of an input and an output words, it first computes the co-occurrence matrix, to avoid training on identical word pairs repetitively.
- Afterwards, it performs matrix decomposition on this co-occurrent matrix.

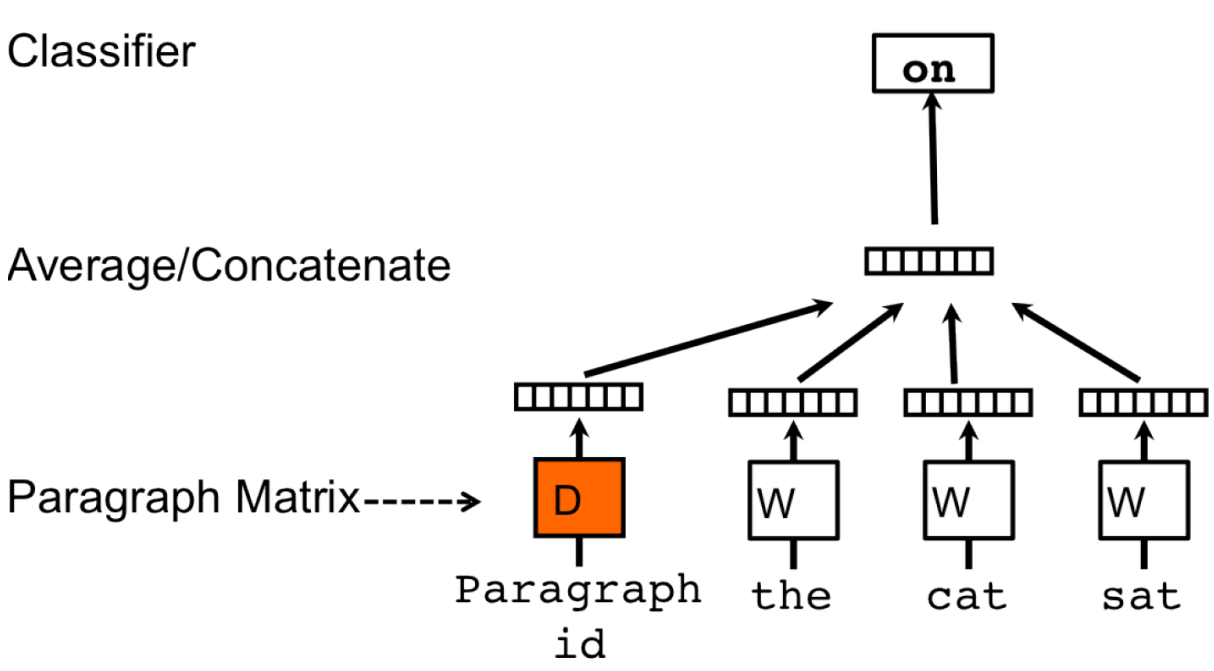
$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2 \quad f \sim$$

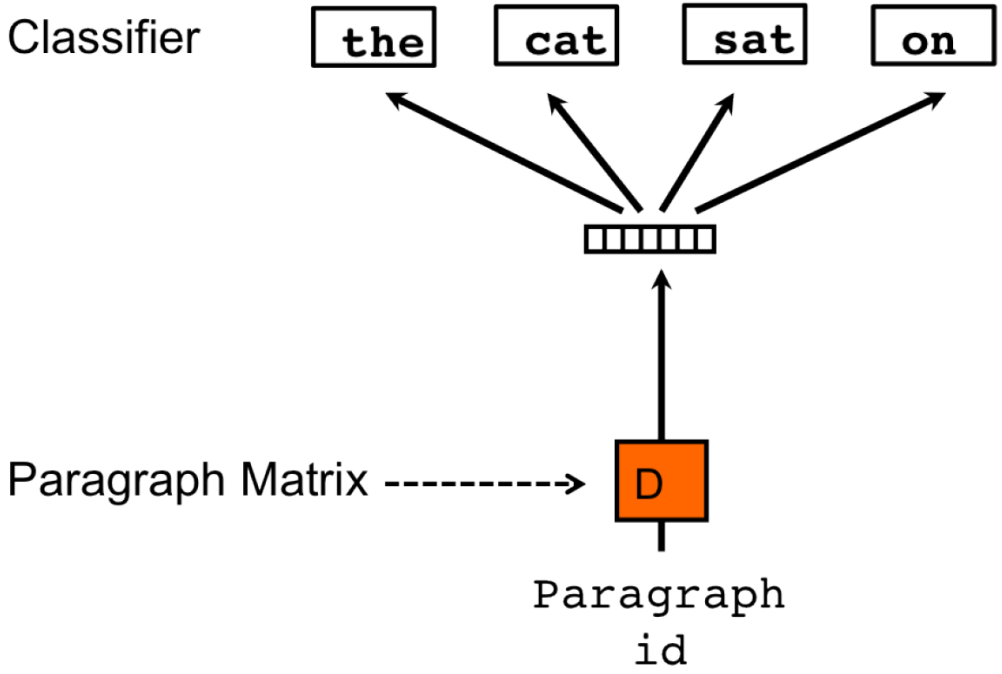


- Fast training
- Works well even with a small corpus

Doc2Vec (Paragraph2Vec)

- Idea: Represent a document (or paragraph) vector as a word
- Properties and Advantages
  - The words in the same paragraphs and documents would have high similarity.
  - A document can be embedded to the same space as a word vector.





Embedding
Sub-word Encoding

Words in writing systems

- Writing systems vary in how they represent words – or don't
- No word segmentation 美国关岛国际机场及其办公室均接获
- Words (mainly) segmented: *This is a sentence with words*
  - Clitics?
    - Separated **Je vous ai apporté** des bonbons
    - Joined فقلناها = ها +نا +قال +ف = so+said+we+it
  - Compounds?
    - Separated life insurance company employee
    - Joined Lebensversicherungsgesellschaftsangestellter

Models below the word level

- Need to handle large, open vocabulary
  - Rich morphology: nejneobhospodařovávatelnějšímu (“to the worst farmable one”)
  - Transliteration: Christopher ↦ Kryštof
  - Informal spelling:

Character-Level Models

- 1. Word embeddings can be composed from character embeddings
  - Generates embeddings for unknown words
  - Similar spellings share similar embeddings
  - Solves out-of-vocabulary (OOV) problem

- 2. Connected language can be processed as characters

- Both methods have proven to work very successfully!
  - Somewhat surprisingly – traditionally, phonemes/letters weren't a semantic unit – but deep learning models can successfully compose groups.

Purely character-level NMT models

- Initially, unsatisfactory performance
  - (Vilar et al., 2007; Neubig et al., 2013)

- Decoder only
  - (Junyoung Chung, Kyunghyun Cho, Yoshua Bengio. arXiv 2016).

- Then promising results
  - (Wang Ling, Isabel Trancoso, Chris Dyer, Alan Black, arXiv 2015)
  - (Thang Luong, Christopher Manning, ACL 2016)
  - (Marta R. Costa-Jussà, José A. R. Fonollosa, ACL 2016)

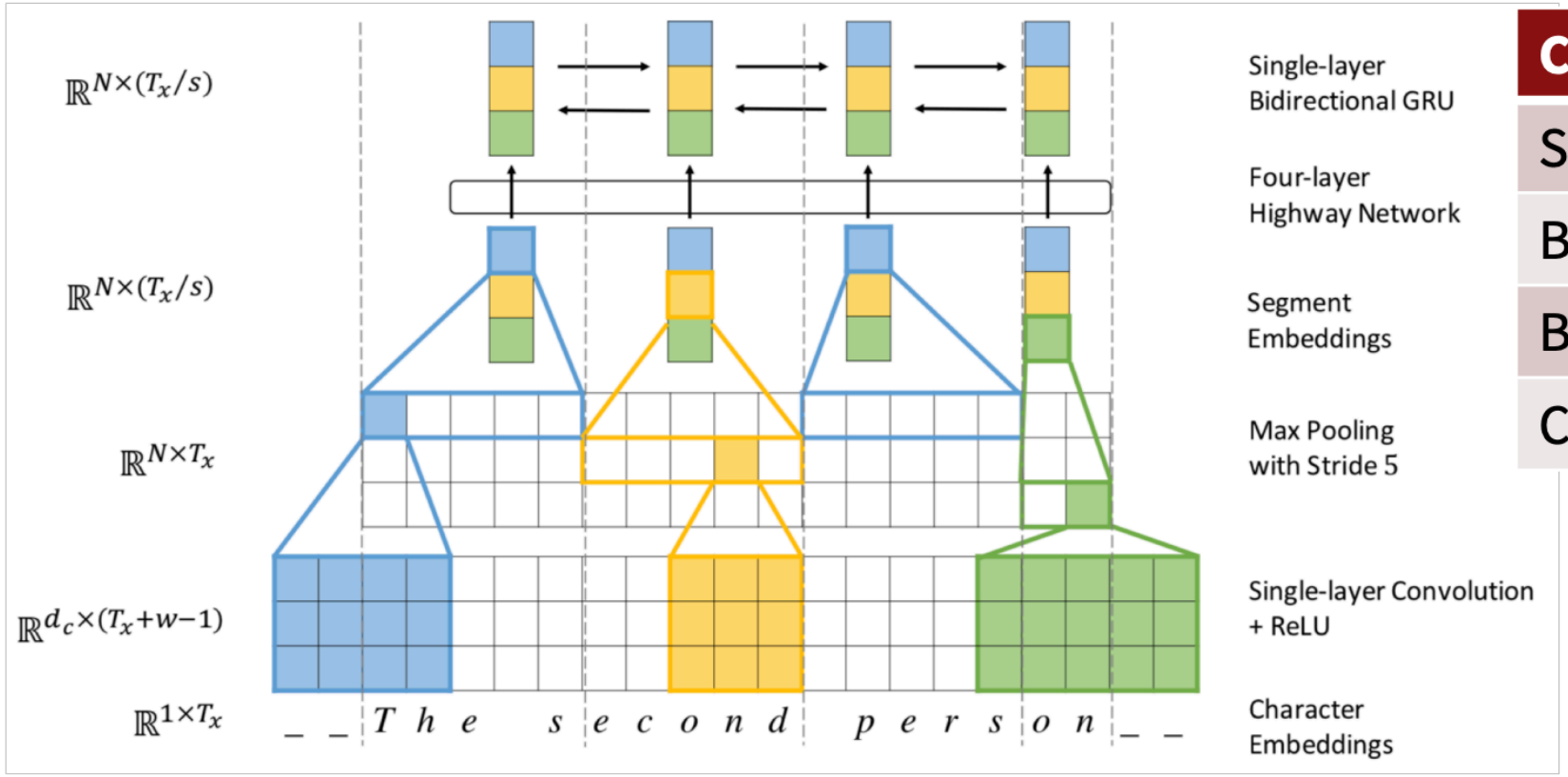
English-Czech WMT 2015 Results

- Luong and Manning tested as a baseline a pure character-level seq2seq (LSTM) NMT system
- It worked well against word-level baseline
- But, it was very slow
  - 3 weeks to train … not that fast at runtime

System	BLEU
Word-level model (single; large vocab; UNK replace)	15.7
Character-level model (single; 600-step backprop)	15.9

Fully Character-Level Neural Machine Translation without Explicit Segmentation

- Jason Lee, Kyunghyun Cho, Thomas Hoffmann. 2017.
- Encoder as below; decoder is a char-level GRU



CS-En	WMT 15	Test
Source	Target	BLEU
Bpe	Bpe	20.3
Bpe	Char	22.4
Char	Char	22.5

Sub-word models: two trends

- Same architecture as for word-level model:
  - But use smaller units: “word pieces”
  - [Sennrich, Haddow, Birch, ACL’16a],
  - [Chung, Cho, Bengio, ACL’16].

- Hybrid architectures
  - Main model has words; something else for characters
  - [Costa-Jussà & Fonollosa, ACL’16],
  - [Luong & Manning, ACL’16].

Byte Pair Encoding

- Originally a compression algorithm:
  - Most frequent byte pair ↦ a new byte.

Replace bytes with character ngrams

(though, actually, some people have done interesting things with bytes)

Rico Sennrich, Barry Haddow, and Alexandra Birch. ***Neural Machine Translation of Rare Words with Subword Units.*** *ACL 2016.*
https://arxiv.org/abs/1508.07909
https://github.com/rsennrich/subword-nmt
https://github.com/EdinburghNLP/nematus

Byte Pair Encoding

- A word segmentation algorithm:
  - Though done as bottom-up clustering
  - Start with a unigram vocabulary of all (Unicode) characters in data
  - Most frequent n-gram pairs ↦ a new n-gram

Byte Pair Encoding

- A word segmentation algorithm
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs ↦ a new n-gram

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d

Start with all characters in vocab

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs ↦ a new n-gram

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs ↦ a new n-gram

Dictionary

5 l o w
2 l o w e r
6 n e w **est**
3 w i d **est**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9

(Example from Sennrich)

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs ↦ a new n-gram

Dictionary

5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7

(Example from Sennrich)

Byte Pair Encoding

- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer (commonly Moses tokenizer for MT)
- Automatically decides vocab for system
  - No longer strongly "word" based in conventional way

Top places in WMT 2016!
Still widely used in WMT 2018

https://github.com/EdinburghNLP/nematus

Wordpiece/Sentencepiece model

- Google NMT (GNMT) uses a variant of this
  - V1: wordpiece model
  - V2: sentencepiece model

- Rather than char n-gram count, uses a greedy approximation to maximizing language model log-likelihood to choose the pieces
  - Add n-gram that maximally reduces perplexity

Wordpiece/Sentencepiece model

- Wordpiece model tokenizes inside words
- Sentencepiece model works from raw text
  - Whitespace is retained as special token (_) and grouped normally
  - You can reverse things at end by joining pieces and recoding them to spaces
  - https://github.com/google/sentencepiece
  - https://arxiv.org/pdf/1804.10959.pdf

Wordpiece/Sentencepiece model

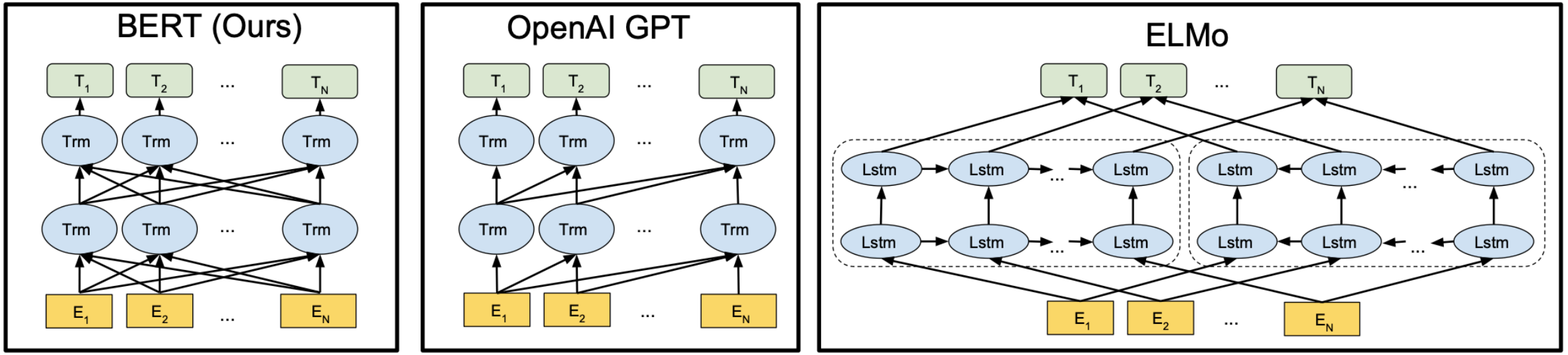
- BERT uses a variant of the wordpiece model
  - (Relatively) common words are in the vocabulary:
    - at, fairfax, 1910s
- Other words are built from wordpieces:
  - hypatia = h ##yp ##ati ##a

- If you're using BERT in an otherwise word based model, you have to deal with this

Embedding
Contextualized Word Embedding

We should consider Context!

ELMo, GPT, BERT



- Token embeddings change depending on the context!
- ELMo: Bidirectional LSTM
- GPT: Transformer Encoder + Masked Attention
- BERT: Transformer Encoder

Natural Language Processing

Edward Choi

Graduate School of AI, KAIST