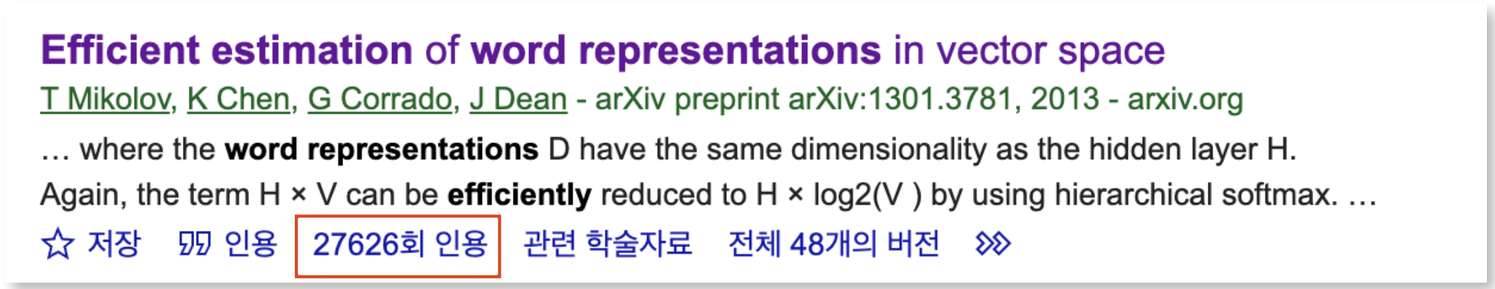
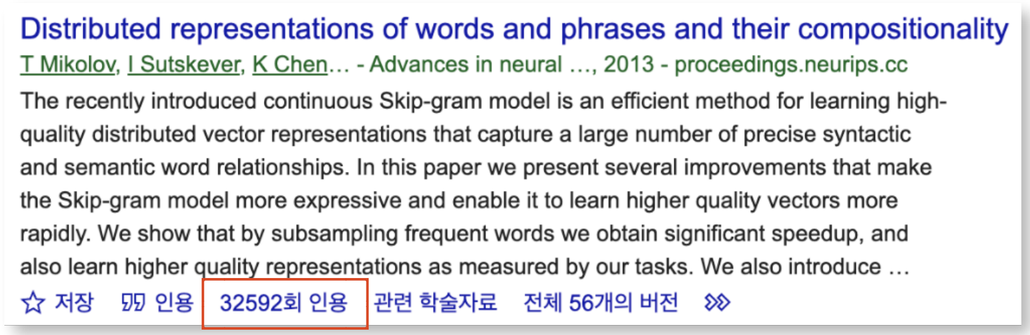
Efficient Estimation of Word Representations in Vector Space (Word2Vec original paper)



KISTI-UST
JUYEON YU

Distributed Representations of Words and Phrases and their Compositionality

(Word2Vec)

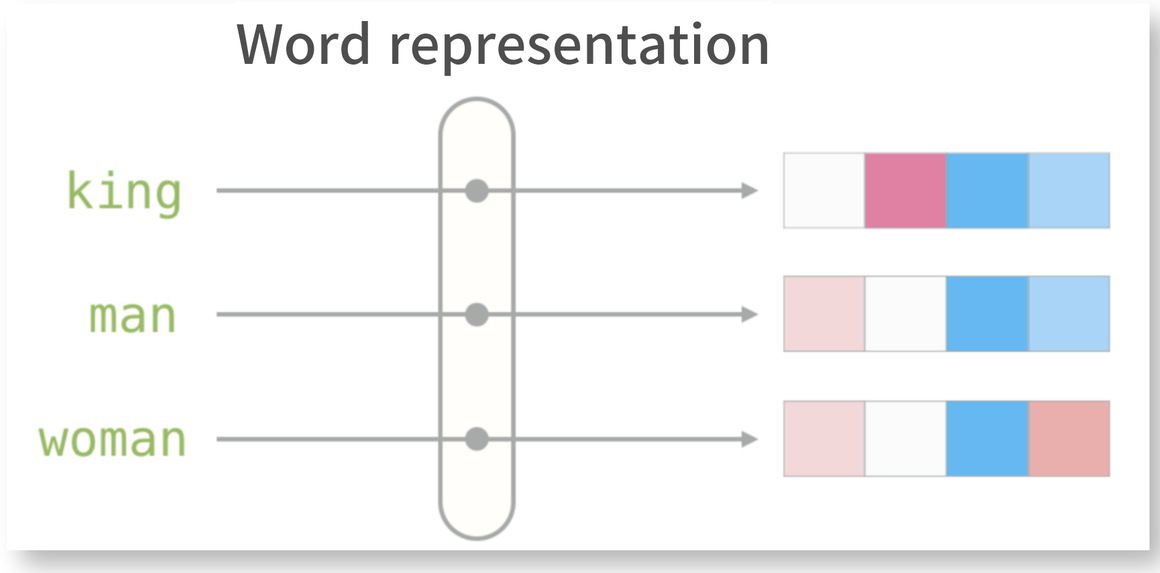


01 Word Representation

What is Word representation?

- A method of representing words as vectors

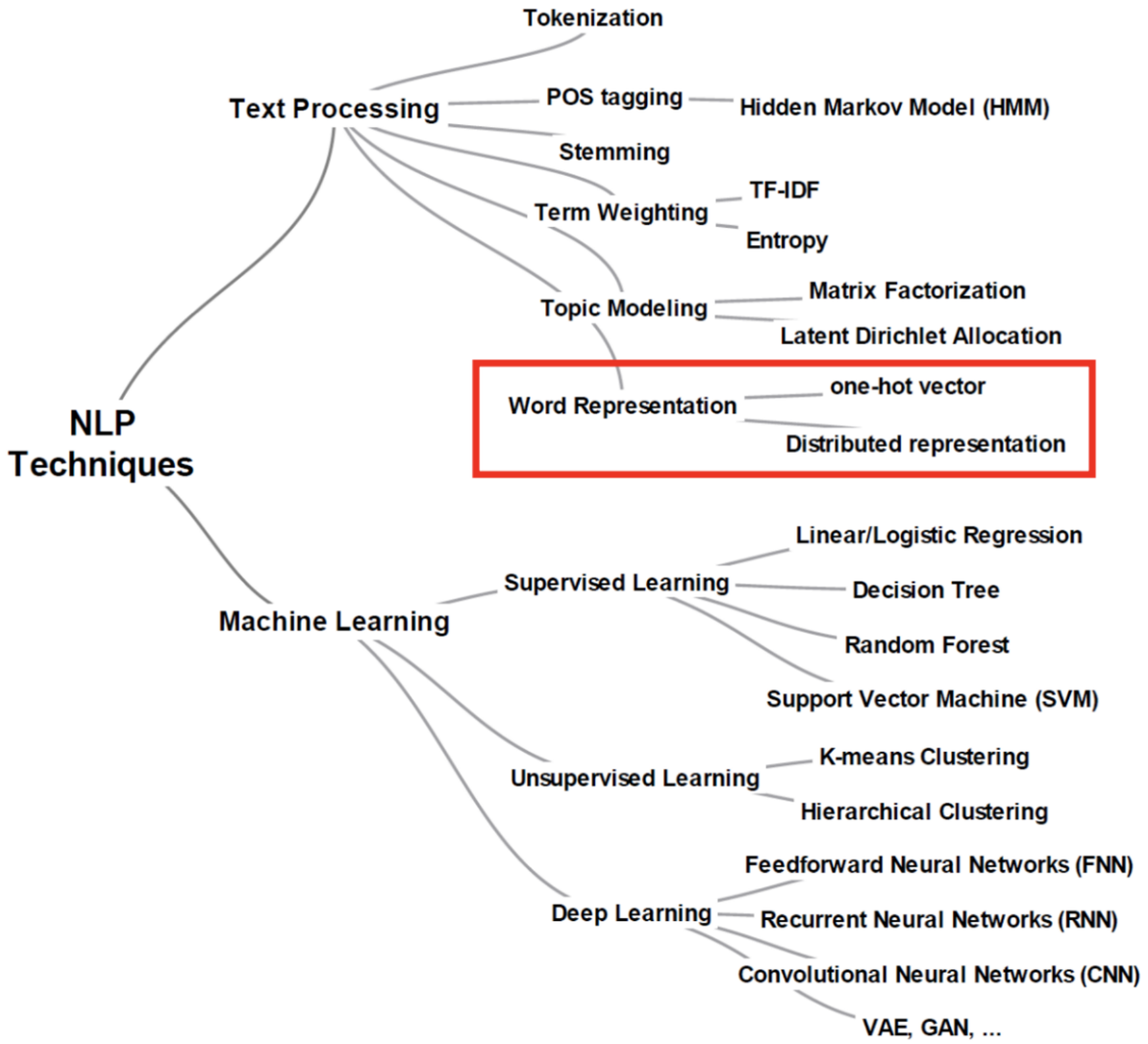
  so that computers can understand and process natural language efficiently.



Source: https://jalammar.github.io/illustrated-word2vec/

01 Word Representation

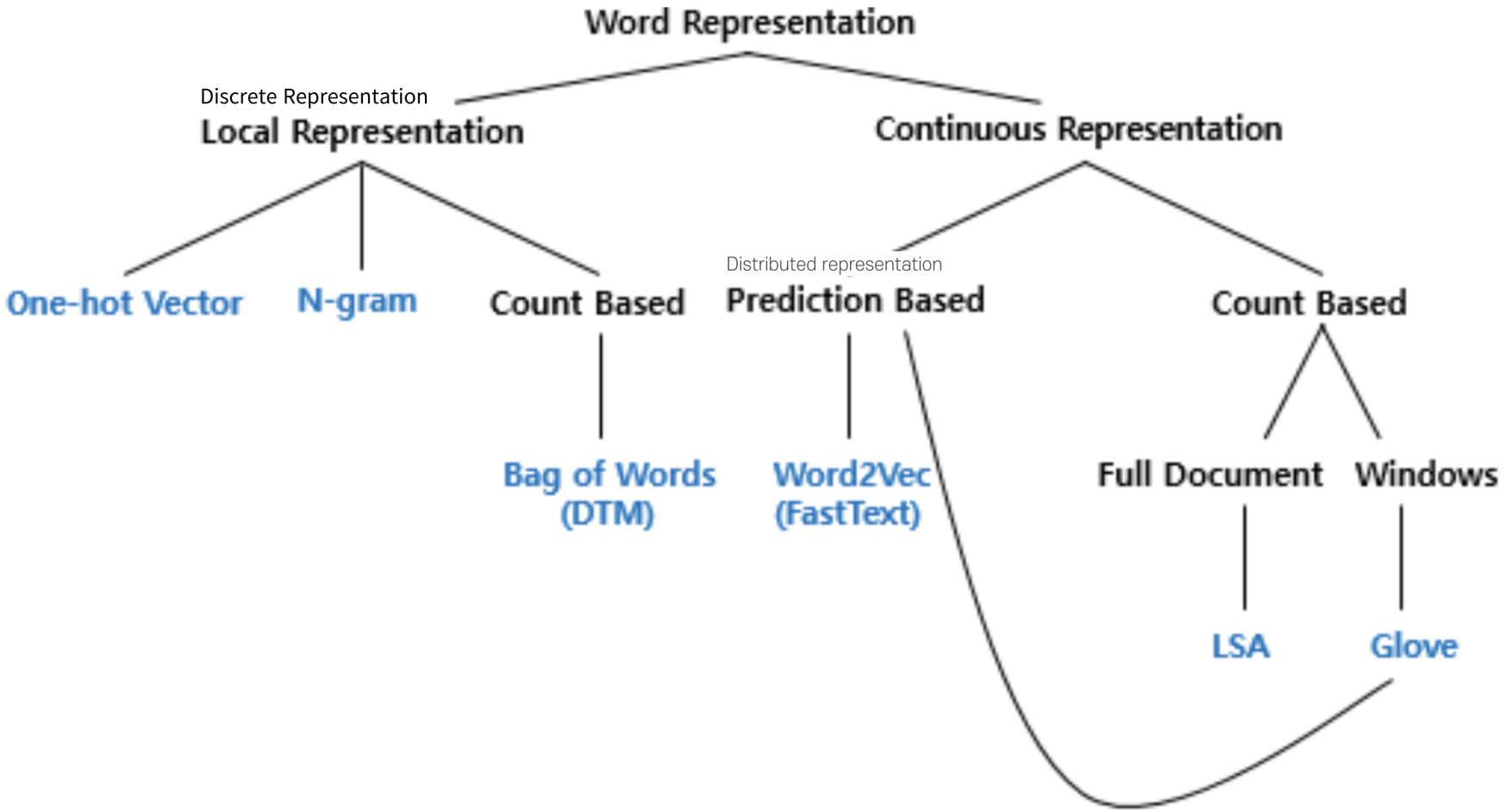
Category of Natrual Language Processing(NLP)



Source: https://statssy.github.io/data/2020/02/06/nlp_ch4/

01 Word Representation

Category of Natrual Language Processing(NLP)



Source: https://statssy.github.io/data/2020/02/06/nlp_ch4/

02 Local Representation

One-hot Vector

- A traditional Natural Language Processing(NLP) method.
- Represent every word as an $\mathbb{R}^{|V|\times 1}$ vector with all 0s and one 1 at the index of that word in the sorted english language.
  - In this notation, |V| is the size of our vocabulary.

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Vocabulary

	1	2	3	...	V
aardvark	1	0	0	0	0
a	0	1	0	0	0
at	0	0	1	0	0
...					
zebra	0	0	0	0	1

- It is simplicity, robustness
- But, it is sparse and cannot capture similarity --> Local Representation

E.g.)

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

⟶ Solution:

Learn to encode similarity in the vectors themselves using

Source: http://web.stanford.edu/class/cs224n

02 Local Representation

How to represent words? Bag of Words

- A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.

- The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.
  - Considering each word count as a feature.

- A bag-of-words is a representation of text that describes the occurrence of words within a document.
  It involves two things:
  - A vocabulary of known words.
  - A measure of the presence of known words.

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

02 Local Representation

Bag of Words - Example

Step 1: Collect Data

Below is a snippet of the first few lines of text from the book "A Tale of Two Cities" by Charles Dickens, taken from Project Gutenberg.

> *It was the best of times,*
> *it was the worst of times,*
> *it was the age of wisdom,*
> *it was the age of foolishness,*

For this small example, let's treat each line as a separate "document" and the 4 lines as our entire corpus of documents.

Step 2: Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary.

The unique words here (ignoring case and punctuation) are:

- "it"
- "was"
- "the"
- "best"
- "of"
- "times"
- "worst"
- "age"
- "wisdom"
- "foolishness"

That is a vocabulary of 10 words from a corpus containing 24 words.

Step 3: Create Document Vectors

The next step is to score the words in each document.

The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.

Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document ("*It was the best of times*") and convert it into a binary vector.

The scoring of the document would look as follows:

- "it" = 1
- "was" = 1
- "the" = 1
- "best" = 1
- "of" = 1
- "times" = 1
- "worst" = 0
- "age" = 0
- "wisdom" = 0
- "foolishness" = 0

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

02 Local Representation

Limitations of Bag-of-Words

▪ Vocabulary: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.

▪ Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.

▪ Meaning: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

As a binary vector, this would look as follows:

```
[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

The other three documents would look as follows:

```
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
```

All ordering of the words is nominally discarded and we have a consistent way of extracting features from any document in our corpus, ready for use in modeling.

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

02 Local Representation

How to represent words? N-gram

- **N-grams are continuous sequences of words or symbols or tokens in a document.**

  E.g.) fine thank you

  - 1-gram (unigram)

    word level : `[fine, thank, you]`

    character level : `[f, i, n, e, , t, h, a, n, k, , y, o, u]`

  - 2-gram (bigram)

    word level : `[fine thank, thank you]`

    character level : `[fi, in, ne, e , t, th, ha, an, nk, k , y, yo, ou]`

  - 3-gram (trigram)

    word level : `[fine thank you]`

    character level : `[fin, ine, ne , e t, th, tha, han, ank, nk , k y, yo, you]`

Source: https://velog.io/@ny_/n-gram

02 Local Representation

Bag of Words vs. N-gram

- **E.g.) machine learning is fun and is not boring**

Bag of Words

machine	fun	is	learning	and	not	boring	...
1	1	2	1	1	1	1	...

- Ignore word order
- Don't know which word "not" modifies.

N-gram

machine learning	learning is	is fun	fun and	and is	is not	not boring	...
1	1	2	1	1	1	1	...

- Can detect typos
- Can predict the next word

Detect typos

When the word qwal comes in,
since the frequency of qu is higher than that of qw, you can correct the typo with qual.

quality, quarter, quit →

Bigram	Count
qu	3
ua	2
al	1
li	1
it	2
ty	1
at	1
te	1
er	1
ui	1

Predict the next word

how are __ →

Trigram	Count
how are you	2
are you doing	1
how are they	1

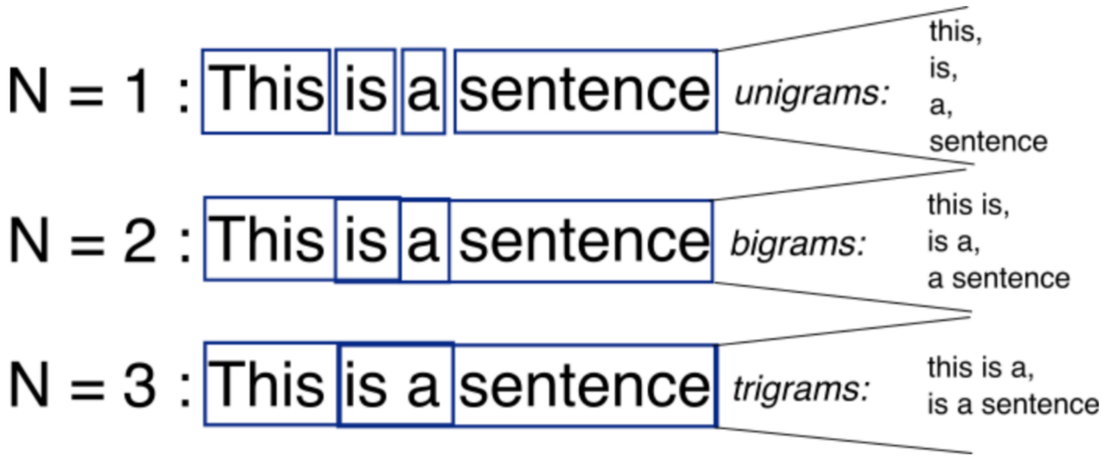
Source: https://velog.io/@ny_/n-gram

02 Local Representation

N-gram language model: more details

- An n-gram is a chunk of n consecutive words.
- Collect statistics about how frequent different n-grams are and use these to predict next word. --> Language Modeling

E.g.) This is a sentence

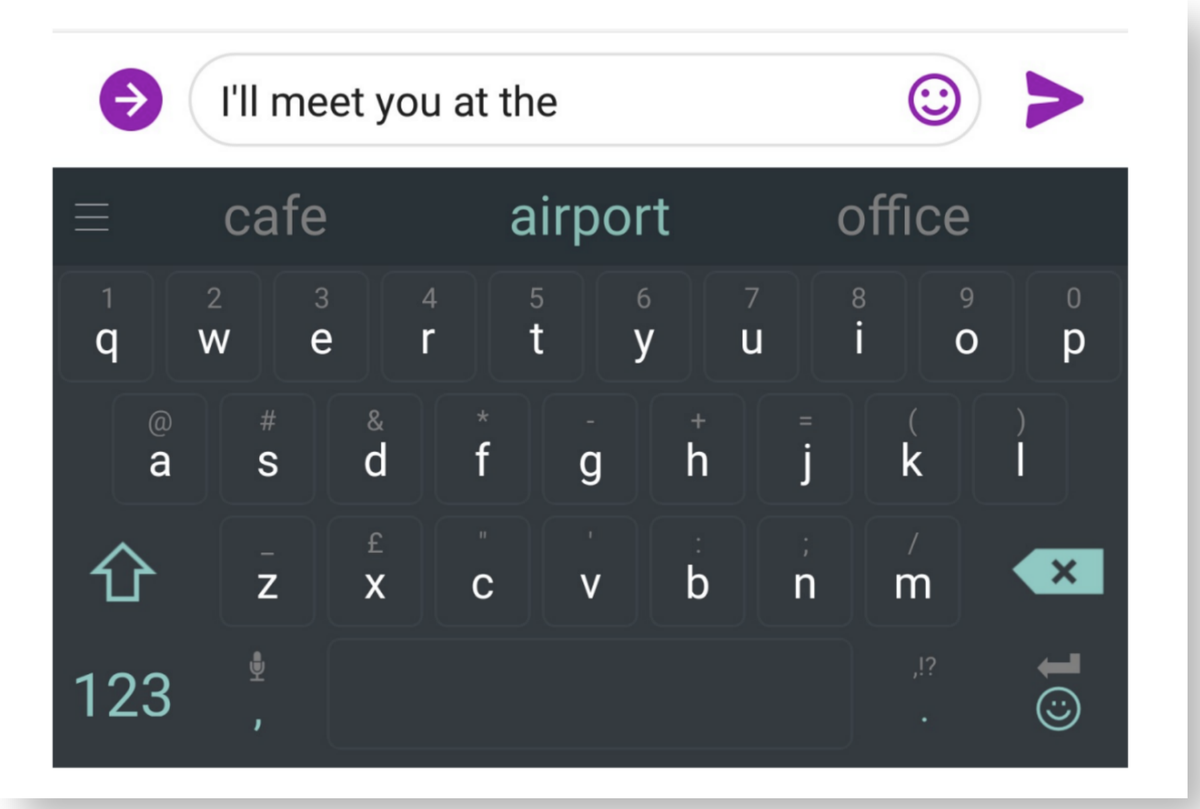


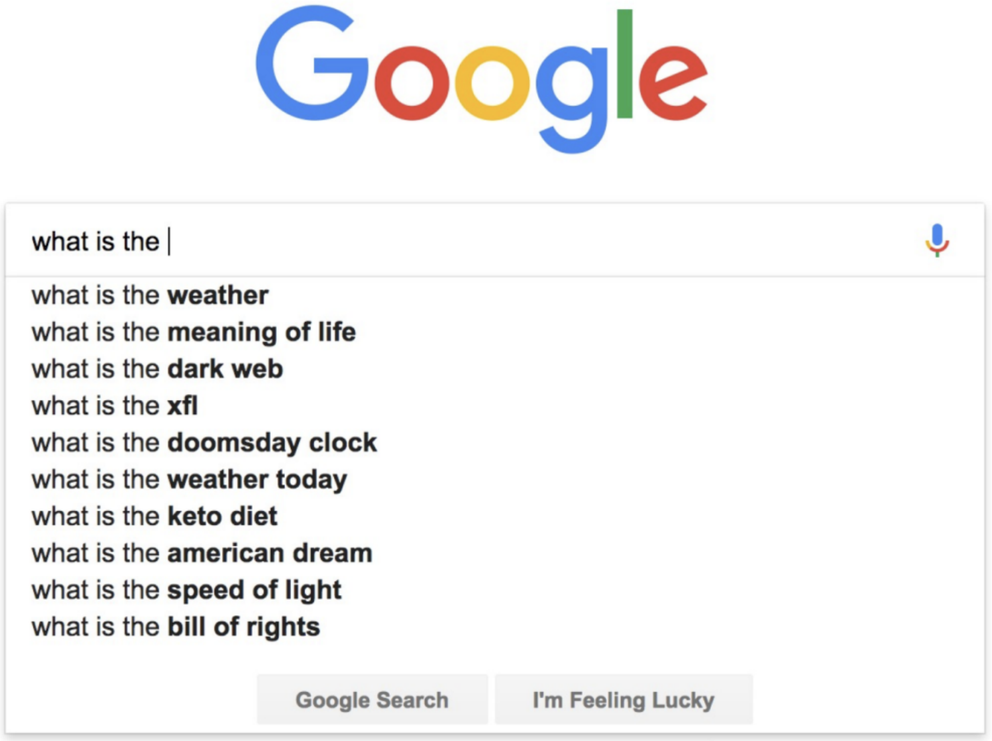
Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

- You use Language Models every day!





Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

- Language Modeling is the task of predicting what word comes next.

  E.g.) the students opened their _____ --> book

  --> laptops

  --> exams

  --> minds

- Given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$,
  compute the probability distribution of the next word : $P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$
  where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

- A system that does this is called a Language Model

Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

- First we make a Markov assumption: $x^{(t+1)}$ depends only on the preceding n-1 words

n-1 words

$$P(x^{(t+1)}|x^{(t)},\ldots,x^{(1)}) = P(x^{(t+1)}|x^{(t)},\ldots,x^{(t-n+2)})$$ Assumption

$$= \frac{P(x^{(t+1)},x^{(t)},\ldots,x^{(t-n+2)})}{P(x^{(t)},\ldots,x^{(t-n+2)})}$$ Definition of conditional prob

$$\approx \frac{\text{count}(x^{(t+1)},x^{(t)},\ldots,x^{(t-n+2)})}{\text{count}(x^{(t)},\ldots,x^{(t-n+2)})}$$ Statistical approximation

Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

- Suppose we are learning a 4-gram Language Model

E.g) ~~as the proctor started the clock~~, the students opened their _____

discard — condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

- For example, suppose that in the corpus:
  - "students opened their" occurred 1000 times
  - "students opened their books" occurred 400 times

    -> P(books | students opened their) = 0.4
  - "students opened their exams" occurred 100 times

    -> P(exams | students opened their) = 0.1

Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

- In some cases, the window of past consecutive n words may not be sufficient to capture the context.

  E.g) as the proctor started the clock, the students opened their _____

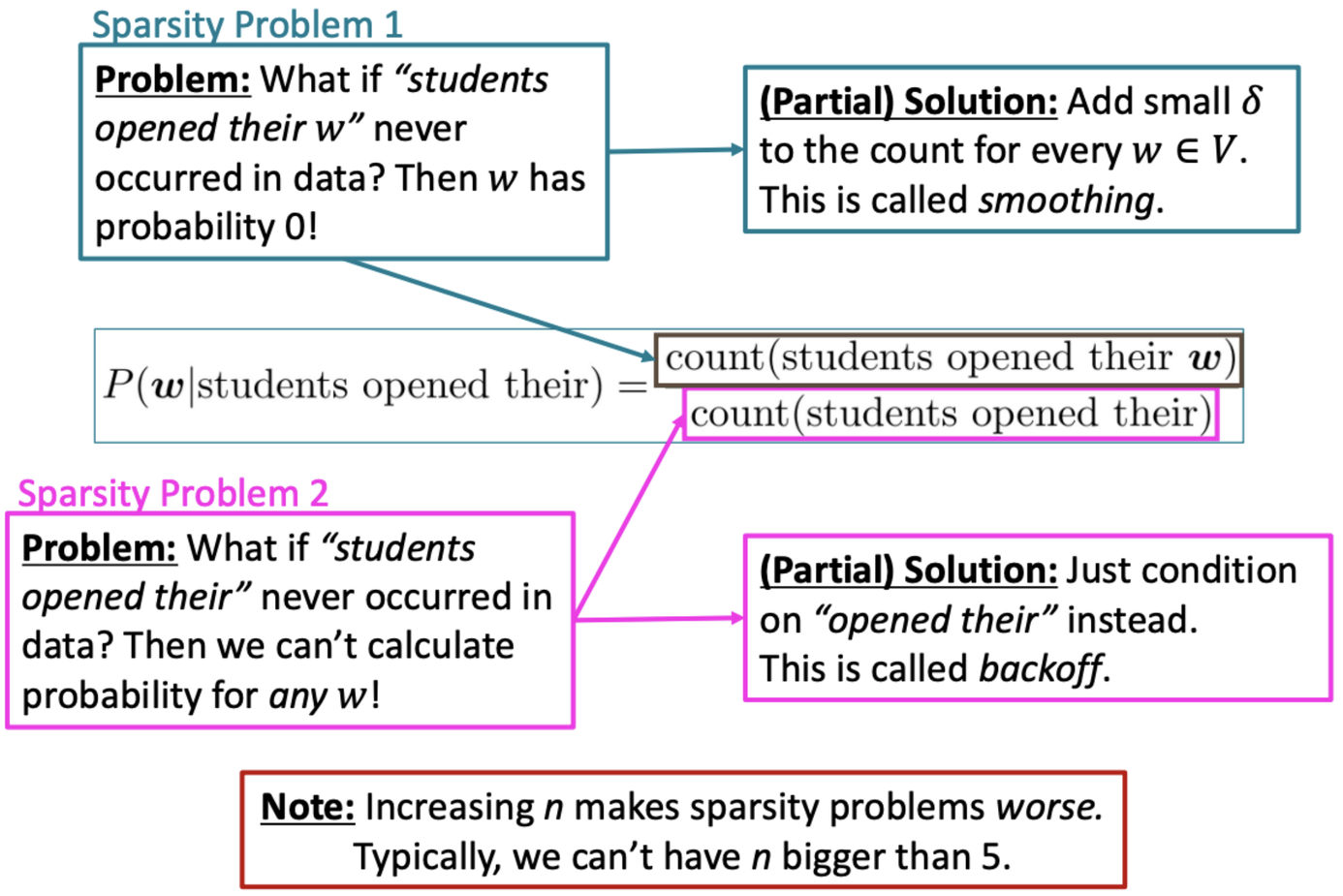
  - If the window only conditions on the previous three words "students opened their",
    the probabilities calculated based on the corpus may suggest that the next word be "books"

  - However, if n had been large enough to include the "proctor" context,
    the probability might have suggested "exam"

- This leads two main issues with n-gram Language Models: Sparsity and Storage.

Source: http://web.stanford.edu/class/cs224n

- Sparsity Problems with n-gram Language Models



Source: http://web.stanford.edu/class/cs224n

- Storage Problems with n-gram Language Models

Storage: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Increasing *n* or increasing corpus increases model size!

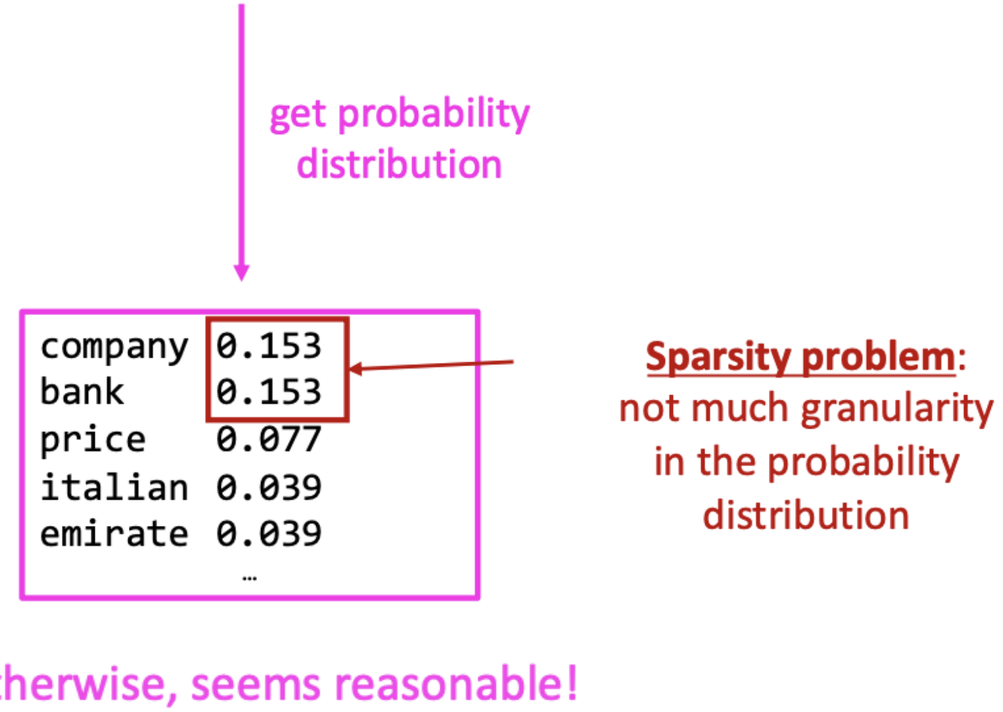
Source: http://web.stanford.edu/class/cs224n

02 Local Representation

N-gram language model: more details

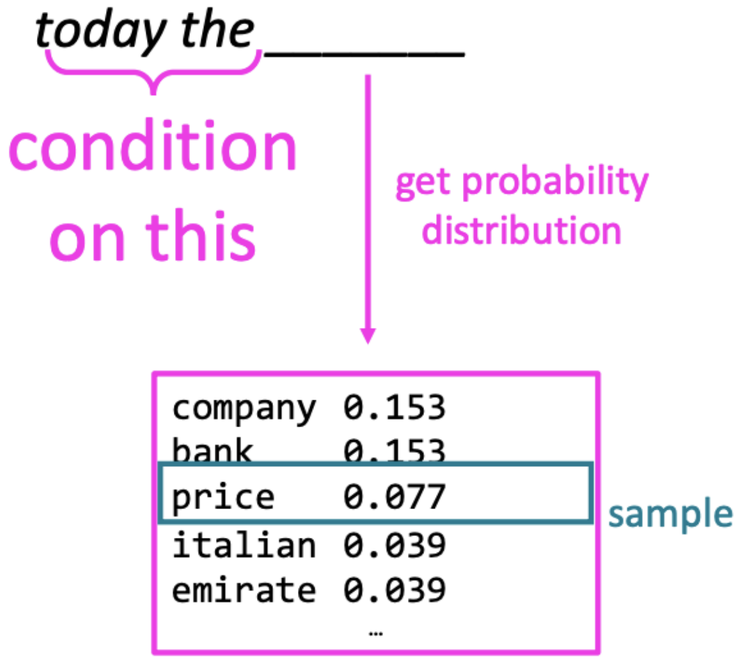
- Suppose we use a Language Model to generate text

today the _____



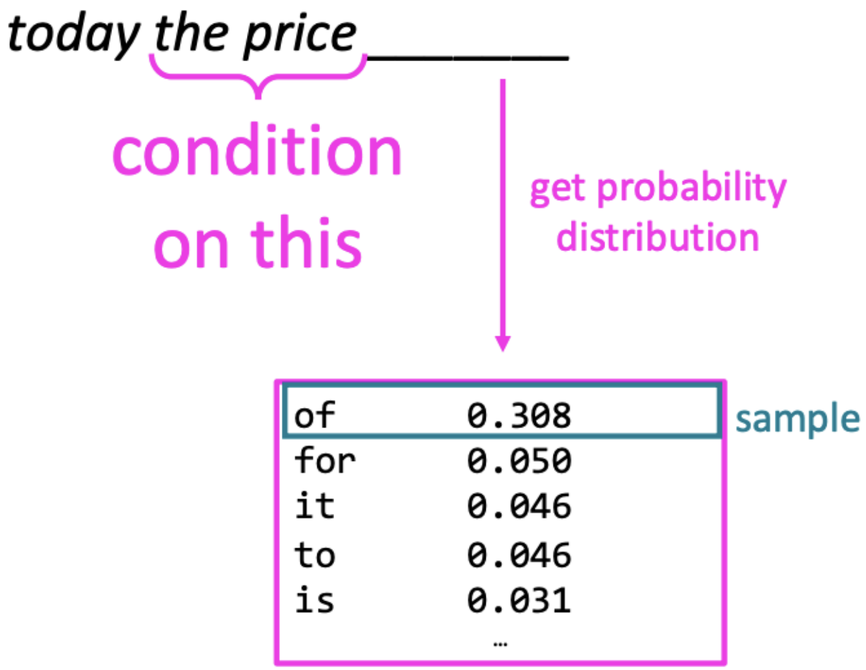
Source: http://web.stanford.edu/class/cs224n

- Suppose we use a Language Model to generate text



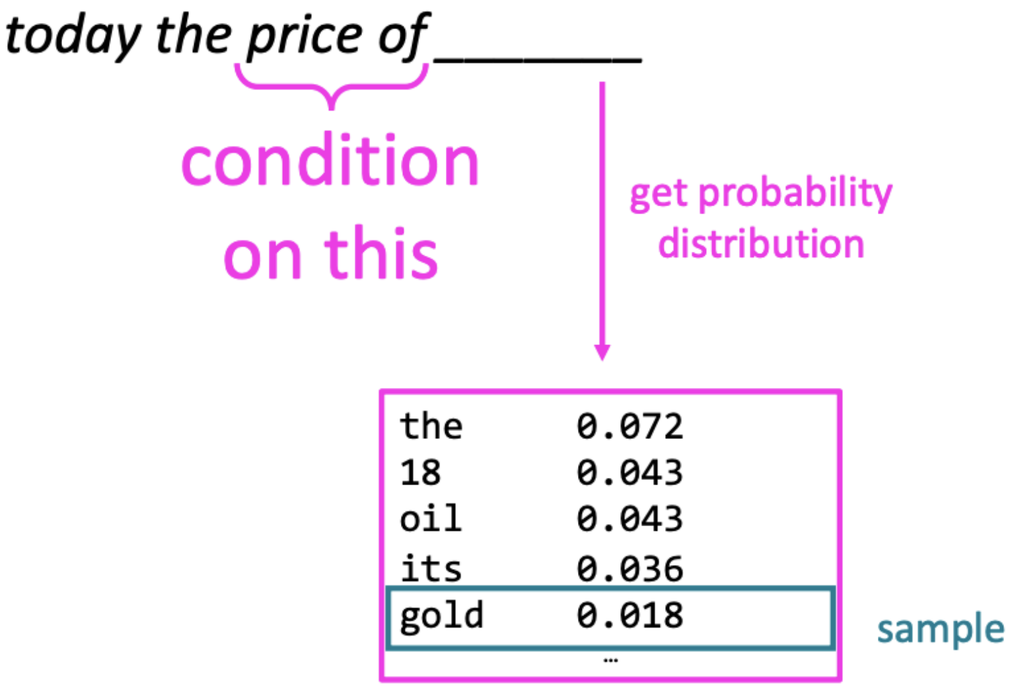
Source: http://web.stanford.edu/class/cs224n

- Suppose we use a Language Model to generate text



Source: http://web.stanford.edu/class/cs224n

- Suppose we use a Language Model to generate text



Source: http://web.stanford.edu/class/cs224n

- Suppose we use a Language Model to generate text

> *today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .*

Surprisingly grammatical!

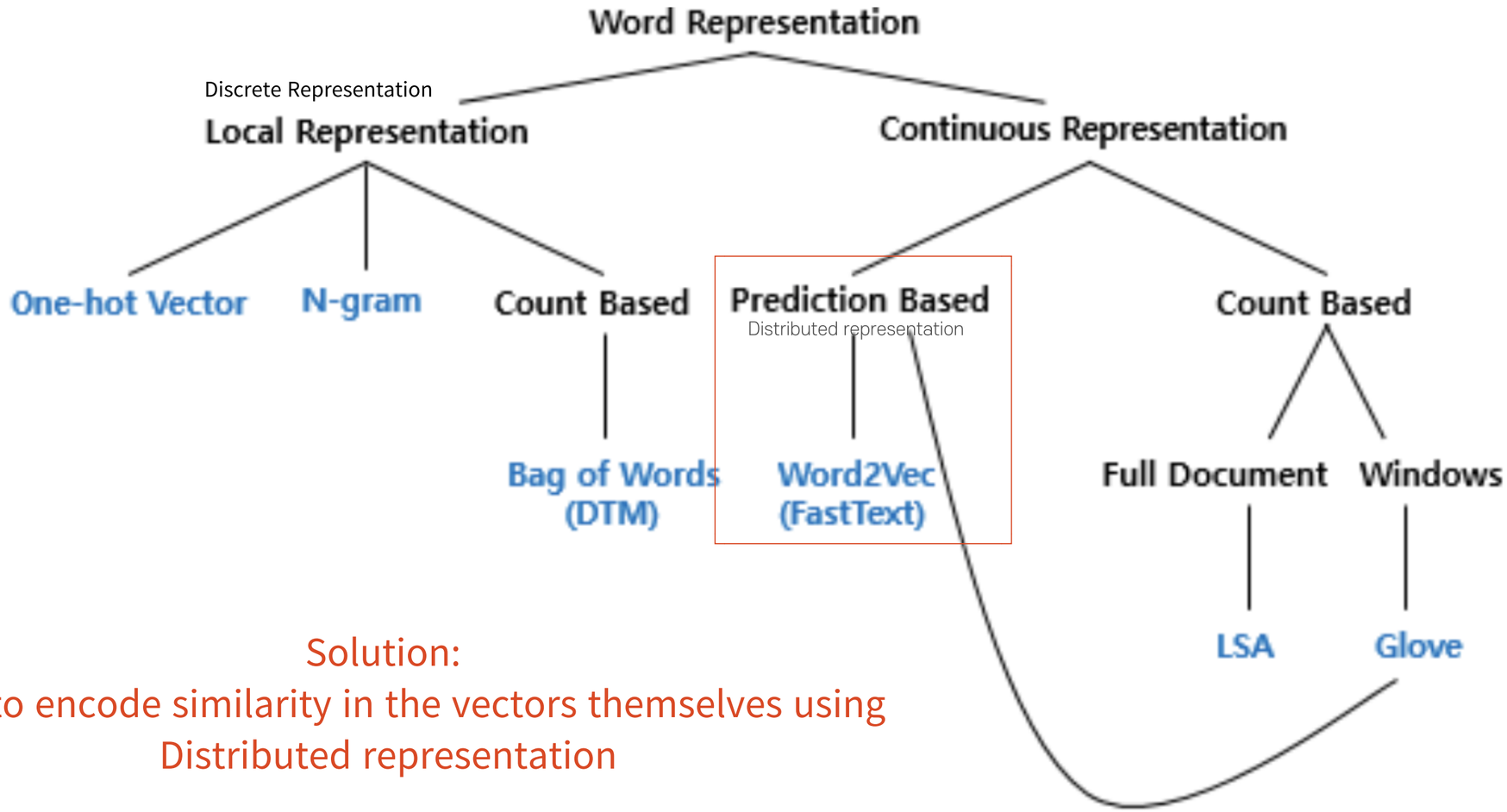
But incoherent. We need to consider more than three words at a time if we want to model language well.

Neural Language Model

Source: http://web.stanford.edu/class/cs224n

03 Continuous Representation

How to represent words? Word2Vec



Solution:
Learn to encode similarity in the vectors themselves using
Distributed representation

Source: https://statssy.github.io/data/2020/02/06/nlp_ch4/

03 Continuous Representation

Distributed representation

- Distributional semantics:
  - A word's meaning is given by the words that frequently appear close-by
  - Representing words by their context.

- When a word *w* appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

- Use the many contexts of *w* to build up a representation of *w*

…government debt problems turning into ***banking*** *crises as happened in 2009…*
…saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge…*
…India has just given its ***banking*** *system a shot in the arm…*

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

Source: http://web.stanford.edu/class/cs224n

03 Continuous Representation

Distributed representation (Word vectors)

- We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

- Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

[Word meaning as a neural word vector]

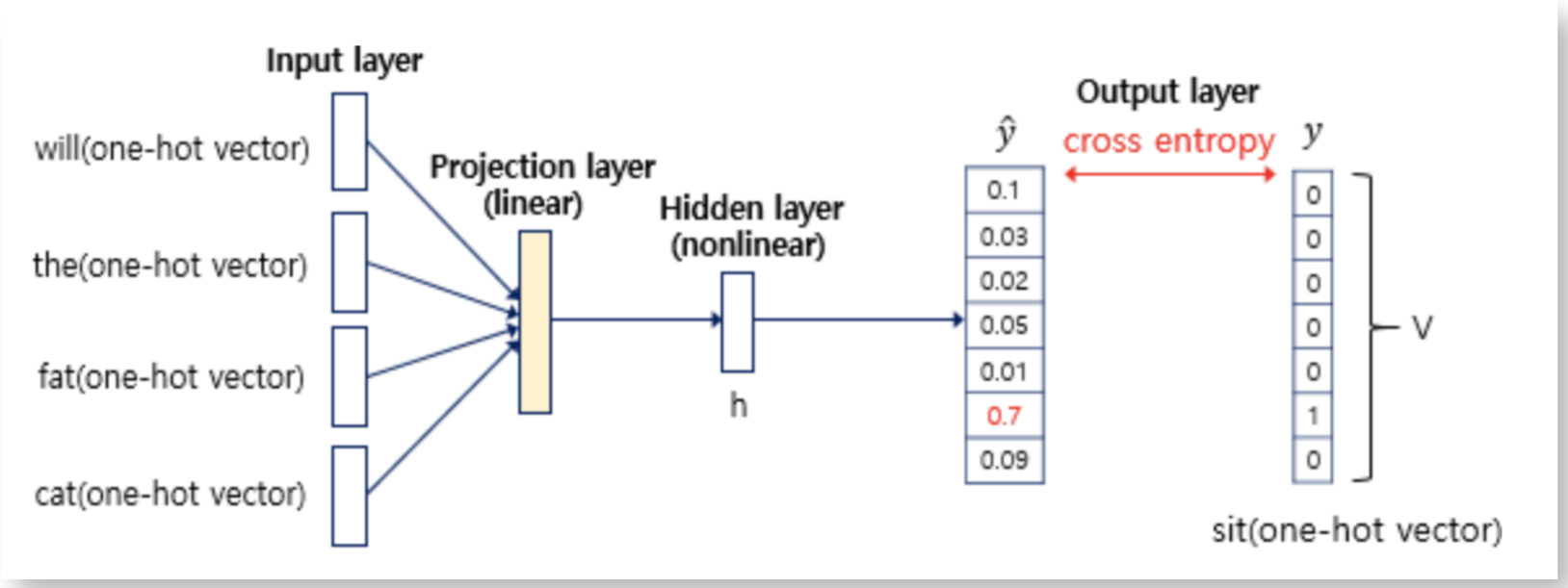
Source: http://web.stanford.edu/class/cs224n

03 Continuous Representation

Neural Network Language Model (NNLM) or Neural Probabilistic Language Model (NPLM)

▪ In 2003, Bengio's paper on NPLM proposes a simple language model architecture which aims at learning a distributed representation of the words in order to solve the curse of dimensionality.

▪ Neural Probabilistic Language Model (NPLM) aims at creating a language model using functionalities and features of artificial neural network.



Source: https://wikidocs.net/45609

03 Continuous Representation

NPLM

✓ Goal : Predicting the target word using the previous n-1 words

"발 없는 말이 천리 간다"

↓ 4-gram

발, 없는, 말이, ??(t=4)

없는, 말이, 천리, ??(t=5)

Step 1 : Initialize C (Shared-Lookup table)

$$C = \begin{bmatrix} 11 & 18 & 25 \\ 10 & 12 & 19 \\ 4 & 6 & 13 \\ 23 & 5 & 7 \\ 17 & 24 & 1 \end{bmatrix} \begin{matrix} \text{발} \\ \text{없는} \\ \text{말이} \\ \text{천리} \\ \text{간다} \end{matrix}$$

Step 2 : Dot product One-hot Vector of each word and C

$$\underset{\text{말이}}{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}} \times \begin{bmatrix} 11 & 18 & 25 \\ 10 & 12 & 19 \\ 4 & 6 & 13 \\ 23 & 5 & 7 \\ 17 & 24 & 1 \end{bmatrix} = \underset{x_{\text{말이}}}{\begin{bmatrix} 23 & 5 & 7 \end{bmatrix}}$$

Step 3 : Concatenate all x

$$x = [x_{t-1}, x_{t-2}, ..., x_{t-n+1}]$$

$$x = [\underbrace{10, 12, 19}_{\text{발}}, \underbrace{4, 6, 13}_{\text{없는}}, \underbrace{23, 5, 7}_{\text{말이}}]$$

Step 4 : $y_{w_t} = b + Wx + Utanh(d + Hx)$

Step 5 : Apply softmax function to y_{w_t}

$$P(w_t|w_{t-1}, ..., w_{t-n+1}) = \frac{exp(y_{w_t})}{\sum_i exp(y_i)}$$

03 Continuous Representation

NPLM

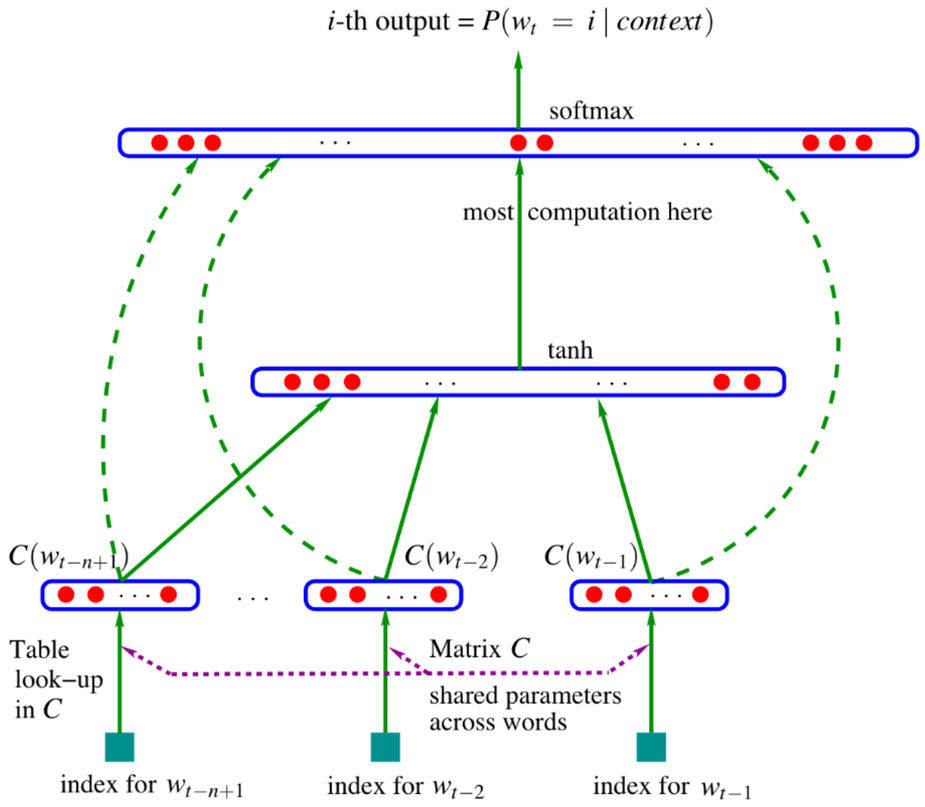


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

✓ Limitation : Too many parameters and computations

$$H \in R^{h \times (n-1)m}, \quad x_t \in R^{(n-1) \times m}, \quad d \in R^{h \times 1}$$
$$U \in R^{|V| \times h}, \quad b \in R^{|V|}, \quad y \in R^{|V|} \quad , C \in R^{m \times |V|}$$

Source: http://web.stanford.edu/class/cs224n

04 Word2vec (Mikolov 2013)

Overview

- Word2vec (Mikolov et al. 2013) is a framework for learning word vectors
- Why two vectors? → Easier optimization. Average both at the end
  - But can implement the algorithm with just one vector per word … and it helps a bit
- Two model variants:
  1. Skip-grams (SG)
     Predict context ("outside") words (position independent) given center word
  2. Continuous Bag of Words (CBOW)
     Predict center word from (bag of) context words

  We presented: **Skip-gram model**

- Additional efficiency in training:
  1. Negative sampling

  So far: Focus on **naïve softmax** (simpler, but expensive, training method)

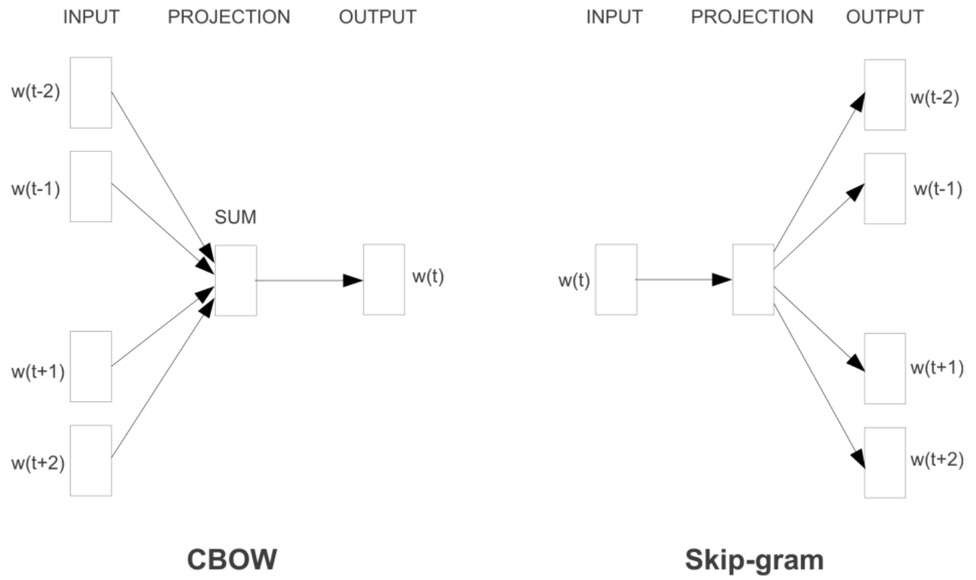


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

04 Word2vec (Mikolov 2013)

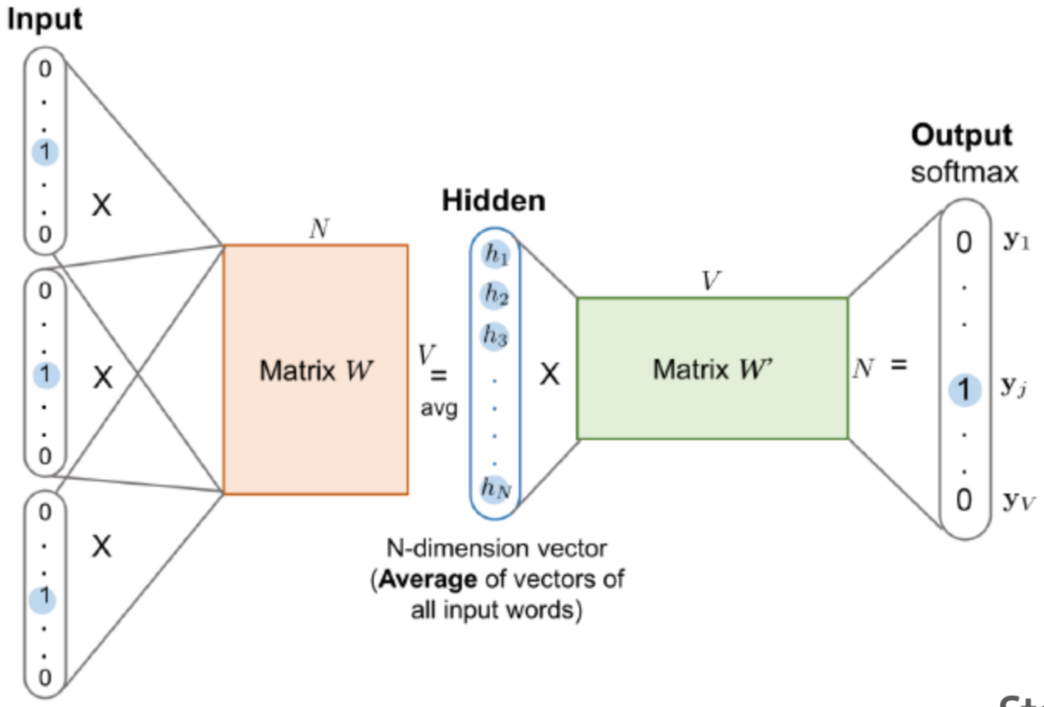
Previous study - CBOW

e.g. "The fat cat sat on the mat", n = 2

X = [fat cat on the], y = sat

✓ CBOW

- Predicting target word using context word
- Goal : Learning word representation



Step 1 :

x_{cat} × $W_{V\times M}$ = V_{cat}

[0 0 1 0 0 0 0] × | 0.5 2.1 1.9 1.5 0.8 | 0.8 1.2 2.8 1.8 2.1 | 2.1 1.8 1.5 1.7 2.7 | = [2.1 1.8 1.5 1.7 2.7]

lookup table

Step 2 :

$$v = \frac{V_{fat} + V_{cat} + V_{on} + V_{the}}{2\times n(winow\ size)}$$

Step 3 : z = v * w′

Step 4 : Y = Softmax(z)

Step 5 : Minimize loss

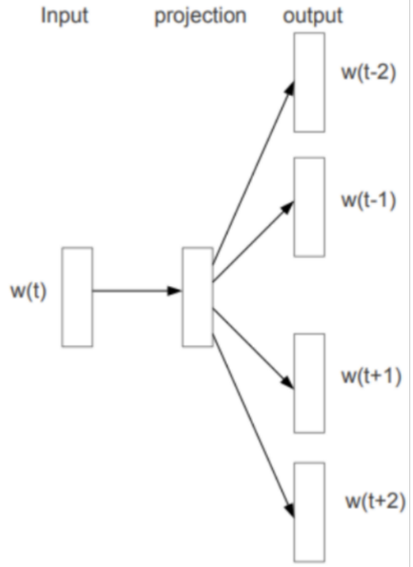
$$minimize J = -\log P(w_c|w_{c-m},\ldots,w_{c+m})$$
$$= -\log P(u_c|v)$$
$$= -\log \frac{exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} exp(u_j^T \hat{v})}$$
$$= -u_c^{intercal}\hat{v} + \log \sum_{j=1}^{|V|} exp(u_j^T \hat{v})$$

Step 6 : Use W, W′ , W|W′ or (W+W′)/2 as word representation

04 Word2vec (Mikolov 2013)

Previous study - Original Skip-gram

- Idea:
  - We have a large corpus of text
  - Every word in a fixed vocabulary is represented by a vector
  - Go through each position t in the text, which has a center word w_I and context ("outside") words w_O
  - Use the similarity of the word vectors for w_I and w_O to calculate the probability of w_O given w_I (or vice versa)
  - Keep adjusting the word vectors to maximize this probability



[Original Skip-gram Model]

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

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top}v_{w_I}\right)}$$

c : The size of the training context
w_t : The center word

w_O : The output word
w_I : The input word
v' : The vector representations of output word
v : The vector representations of input word
W : The number of words in vocabulary

Complexity : $O(V)$

04 Word2vec (Mikolov 2013)

NPLM vs. CBOW vs. Original Skip-gram

✓ The number of parameters is reduced.

$$H \in R^{h \times (n-1)m}, \quad x_t \in R^{(n-1) \times m}, \quad d \in R^{h \times 1}$$
$$U \in R^{|V| \times h}, \quad b \in R^{|V|}, \quad y \in R^{|V|}, \quad C \in R^{m \times |V|}$$

[NPLM]

$$W \in R^{|V| \times N}, W' \in R^{N \times |V|}$$

[CBOW and Skip-gram]

$2*w \times N + N \times V$

[CBOW]

$(1 \times N + N \times V)*2*w$

[Skip-gram]

✓ But it's still too expensive because of the softmax function. (Complexity O(V))

04 Word2vec (Mikolov 2013)

Hierarchical Softmax

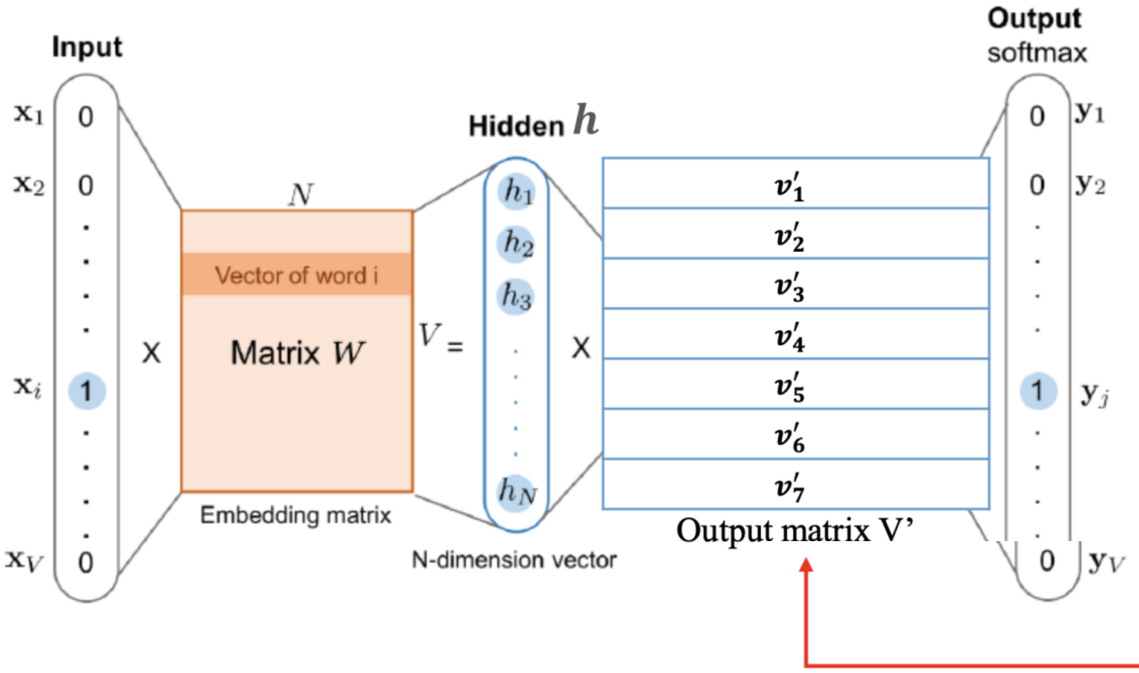
✓ Goal : To computationally efficient approximate the full softmax using binary tree.

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I}\right)$$

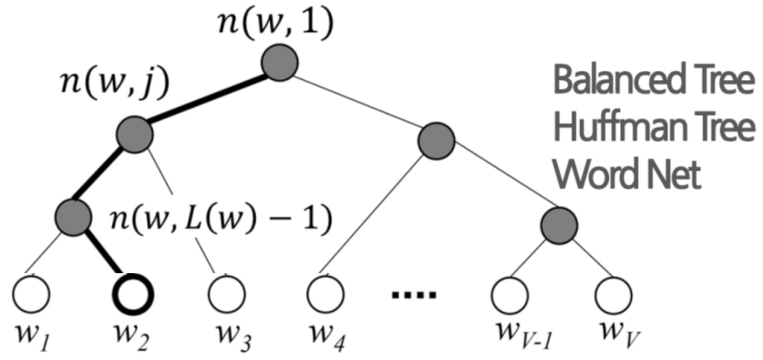
[Objective Function]

h

$n(w,j)$: The j-th node on the path from the root to w
$L(w)$: The length of above path
$n(w,1)$: The root
$n(w,L(w))$: w
$ch(n)$: The left child of n
$[\![x]\!]$: 1 if x is true (left), -1 otherwise
$\sigma(x) = \frac{1}{1+exp(-x)}$: Sigmoid Function
$\sum_{w=1}^{W} p(w|w_i) = 1$



Complexity : $O(log_2 V)$



04 Word2vec (Mikolov 2013)

Negative Sampling

- ✓ Goal : To computationally efficient approximate the full softmax
- ✓ Change problem setting multi classification to binary classification (+, -)

For Positive Sample ↓ For Negative Sample ↓

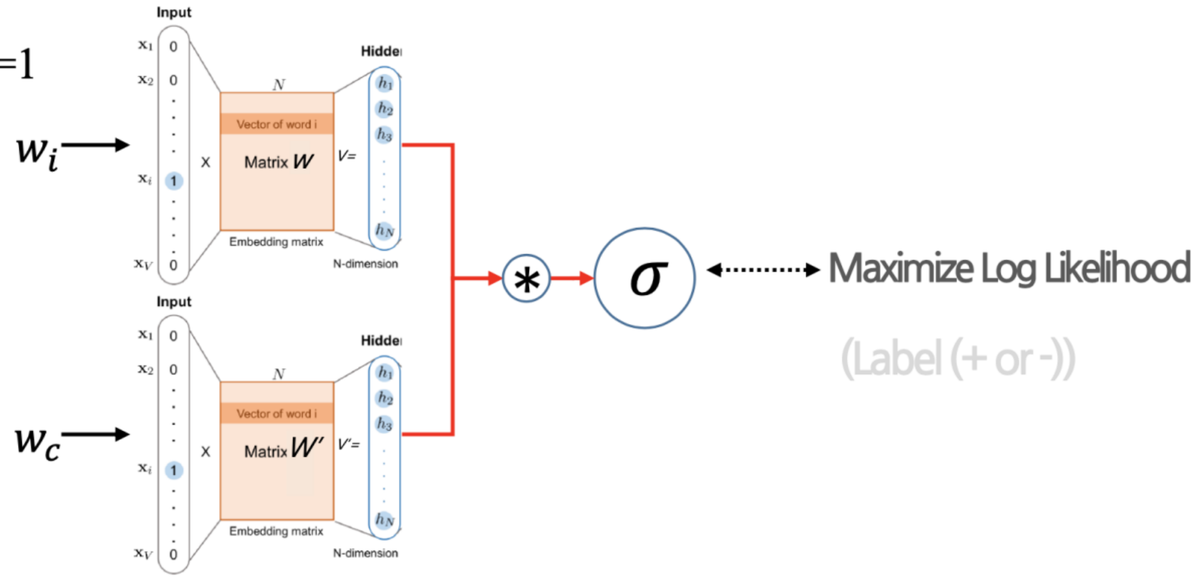
$$\log \sigma(v'_{w_O}{}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-v'_{w_i}{}^{\top} v_{w_I}) \right]$$

k : The number of negative sample (5-20 for small datasets, 2-5 for large datasets)

e.g. "The fat cat sat on the mat that can be tee of toy ", n=2, k=1

	[Positive Set]	[Negative Set]
(x, y) =	(sat, fat),	
	(sat, cat),	
	(sat, on),	
	(sat, the)	
(x1, x2) =	(sat, fat),	(sat, tee),
	(sat, cat),	(sat, toy),
	(sat, on),	(sat, mat),
	(sat, the)	(sat, can)

[Original Skip-gram] [Skip-gram with Negative Sampling]



Word2vec (Mikolov 2013)

Subsampling of Frequent Words

✓ Goal : To counter the imbalance between the rare and frequent words

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}(f(w_j)^{3/4})}$$

e.g. “A, A, B”

$$f(A) = p(A) = \frac{2}{3} \qquad f(B) = p(B) = \frac{1}{3}$$

$$P(A) = \frac{\left(\frac{2}{3}\right)^{\frac{3}{4}}}{\left(\frac{2}{3}\right)^{\frac{3}{4}} + \left(\frac{1}{3}\right)^{\frac{3}{4}}} = 0.6271$$

$$P(B) = \frac{\left(\frac{1}{3}\right)^{\frac{3}{4}}}{\left(\frac{2}{3}\right)^{\frac{3}{4}} + \left(\frac{1}{3}\right)^{\frac{3}{4}}} = 0.3729$$

[The probability of Negative sampling]

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

t : Hyper Parameter (10^{-5})

- Calculate the probability of being excluded in training

e.g. “A, A, B”

$$f(A) = p(A) = \frac{2}{3} \qquad f(B) = p(B) = \frac{1}{3}$$

$$P(A) = 1 - \sqrt{\frac{10^{-5}}{\left(\frac{2}{3}\right)}} = 0.9961$$

$$P(B) = 1 - \sqrt{\frac{10^{-5}}{\left(\frac{1}{3}\right)}} = 0.9945$$

[The probability of Subsampling]

04 Empirical Results

✓ To evaluate the Hierarchical Softmax (HS), Noise Contrastive Estimation, Negative Sampling, and subsampling of the training words, the analogical reasoning task is used.

- Datasets : Large dataset consisting of various news articles(an internal Google dataset with one billion words)
  : All words that occurred less than 5 times in the training data is discarded from the vocabulary.
  (Final vocabulary of size 692K)

Ex) Vec("Berlin") - Vec("Germany") + Vec("France") = Vec("Paris")

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Table 1: Accuracy of various Skip-gram 300-dimensional models on the analogical reasoning task as defined in [8]. NEG-k stands for Negative Sampling with k negative samples for each positive sample; NCE stands for Noise Contrastive Estimation and HS-Huffman stands for the Hierarchical Softmax with the frequency-based Huffman codes.

04 Learning Phrases

- ✓ To learn vector representation for phrases, this paper first find words that appear frequently together, and infrequently in other contexts.
- ✓ For example, "New York Times" and "Toronto Maple Leafs" are replaced by unique tokens in the training data, while a bigram "this is" will remain unchanged.

Method	Dimensionality	No subsampling [%]	10^{-5} subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	19	**47**

Table 3: Accuracies of the Skip-gram models on the phrase analogy dataset. The models were trained on approximately one billion words from the news dataset.

	NEG-15 with 10^{-5} subsampling	HS with 10^{-5} subsampling
Vasco de Gama	Lingsugur	Italian explorer
Lake Baikal	Great Rift Valley	Aral Sea
Alan Bean	Rebbeca Naomi	moonwalker
Ionian Sea	Ruegen	Ionian Islands
chess master	chess grandmaster	Garry Kasparov

Table 4: Examples of the closest entities to the given short phrases, using two different models.

04 Additive Composionality

- ✓ This paper found that the Skip-gram representations exhibit another kind of linear structure that makes it possible to meaningfully combine words by an element-wise addition of their vector representations.

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

04 Comparison to Published Word Representations

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

05 Conclusion

Recap

- ✓ This work show how to train distributed representations of words and phrases with the Skip-gram model and demonstrate that these representations exhibit linear structure that makes precise analogical reasoning possible.
- ✓ With the same size of datasets, word representation can be more efficiently trained.
- ✓ Hierarchical Softmax
- ✓ Negative Sampling
- ✓ Subsampling of Frequent Words

05 Conclusion

Towards GloVe: Count based 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)

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

https://web.stanford.edu/class/archive/cs/cs224n/

THANK YOU

Q & A