Chapter 04. 자연어처리 (Natural Language Processing)

# 단어를 숫자로 표현하기

# 컴퓨터가 보는 문자

## ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

컴퓨터는 ASCII, 유니코드, UTF-8 encoding 등으로 문자를 표현하고 저장한다.

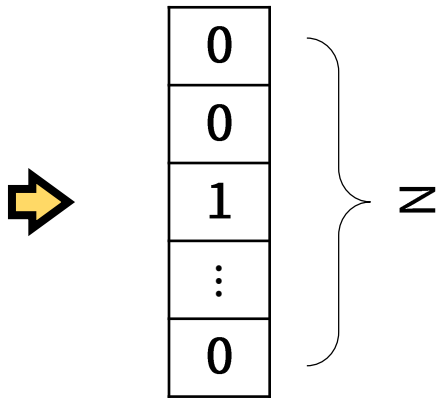
# 컴퓨터가 보는 단어

| 단어 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| love | 0x6C | 0x6F | 0x76 | 0x65 |
| live | 0x6C | 0x69 | 0x76 | 0x65 |
| like | 0x6C | 0x69 | 0x6B | 0x65 |

love는 live보다 like와 더 유사하다. 하지만 컴퓨터가 보기에는 love는 live와 더 비슷해 보인다.

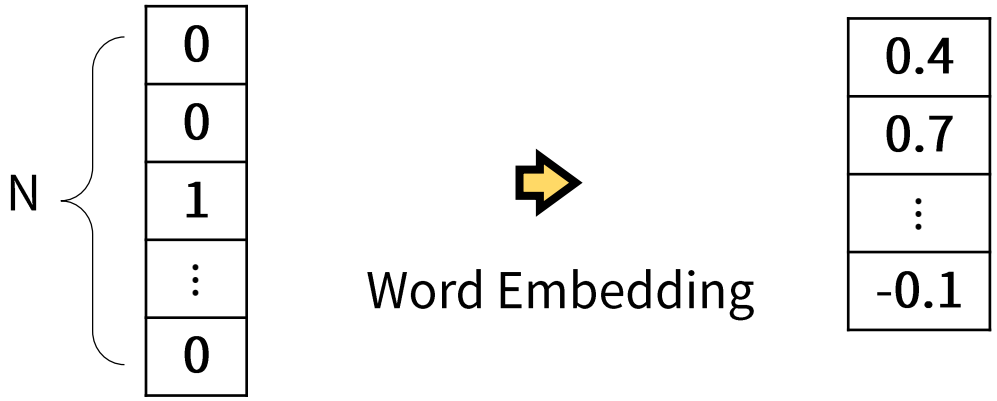
# One-Hot Encoding

| Code | Word |
|---|---|
| 0 | NOT USED |
| 1 | i |
| 2 | you |
| 3 | study |
| 4 | fast |
| 5 | campus |
| … | … |
| N-1 | zzz |




N개의 단어를 좌측의 코드로 표현하면 희소 표현(Sparse representation)이라고 하며,
우측의 벡터로 표현할 경우 One-Hot Encoding이라고 한다.

# 밀집 표현 Dense Representation




희소 표현된 단어를 임의의 길이의 실수 벡터로 표현할 경우, 이를 밀집 표현(Dense Representation)이라고 한다.

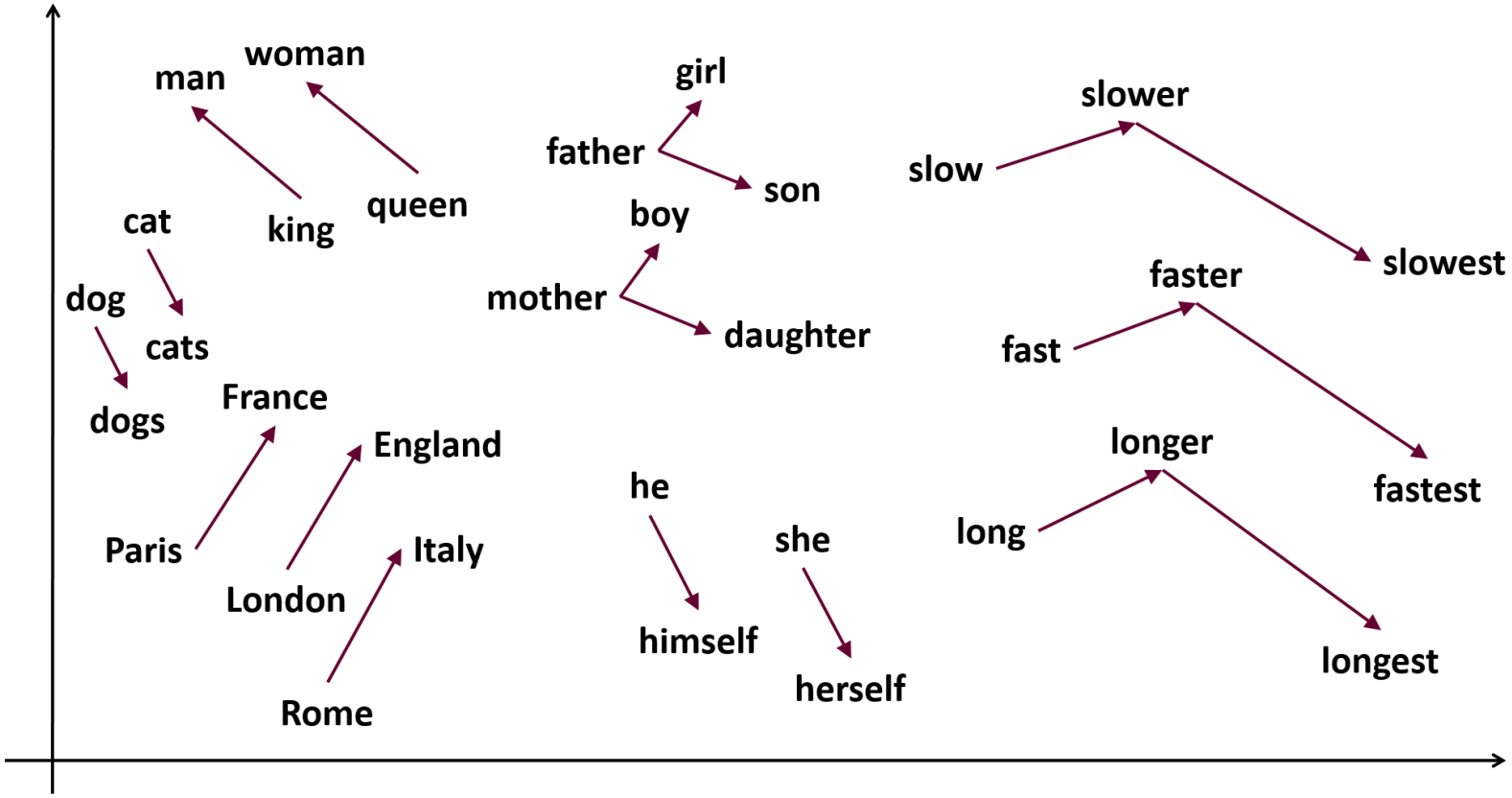
이 과정을 Word Embedding이라고 하며, 밀집 표현된 결과를 임베딩 벡터(Embedding Vector)라고 부른다.

# 말뭉치 Corpus


특정 목적을 가진 언어의 표본. 분석의 용이성을 위해 형태소 분석이 포함되기도 한다.

언어학 연구에 쓰이는 확률/통계적인 자료이며, 딥러닝에도 유용하게 쓰인다.

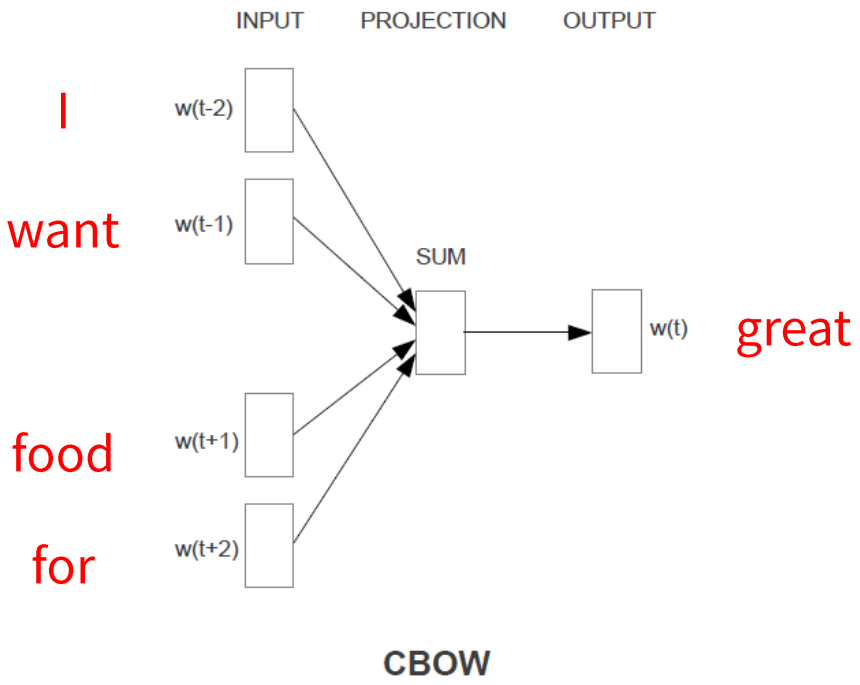
# Word2Vec




Word2Vec은 가장 많이 사용되는 Word Enbedding 방법이다.

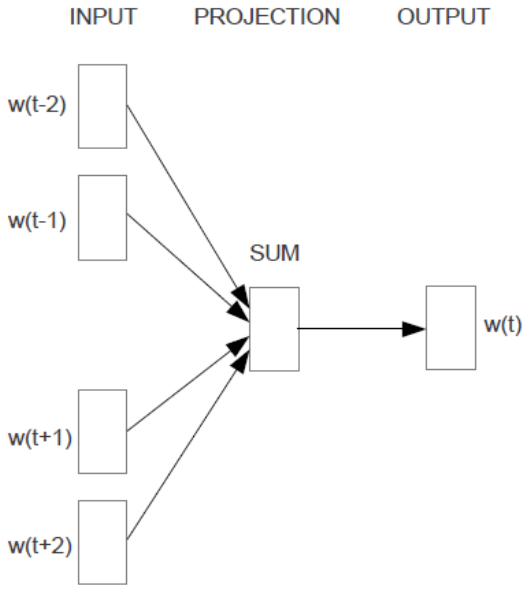
Word2Vec은 유사한 의미를 가진 단어는 유사한 벡터가 되는 특징이 있다.

# CBOW Continuous Bag-of-Words




CBOW 모델은 주변의 단어로 현재 단어를 추정하는 방법이다(Window size=2).

# CBOW의 수식




CBOW

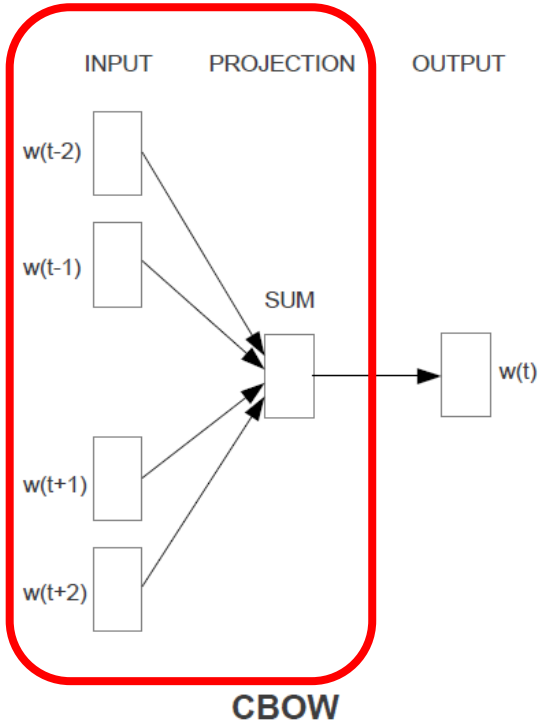
$$\boldsymbol{v} = \frac{\sum_{i=1}^{|n|} W \boldsymbol{x}_{t+i}}{2n}$$

$$\boldsymbol{y} = \text{softmax}(W'\boldsymbol{v})$$

크게 Projection layer와 Output layer로 구분된다.

# Projection Layer

$$v = \frac{\sum_{i=1}^{|n|} W x_{t+i}}{2n}$$




$$\begin{bmatrix} 0.4 & -0.1 & 0.1 \\ 0.2 & 0.6 & 0.0 \\ -0.5 & -0.3 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

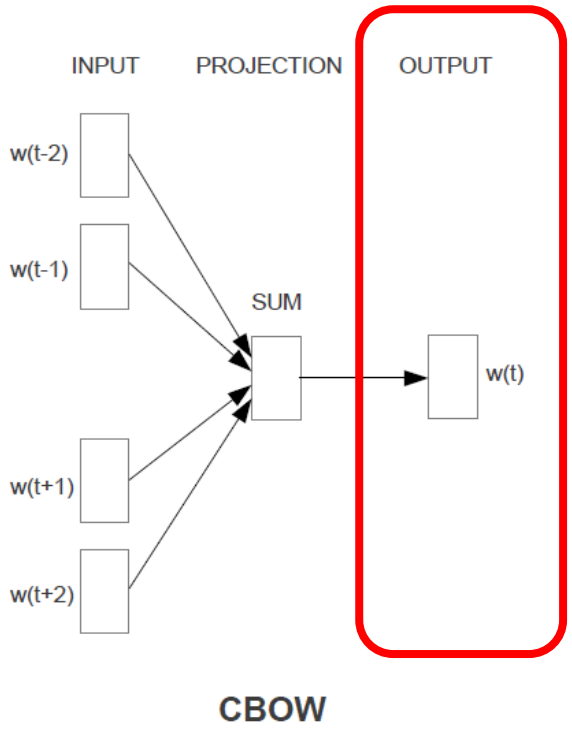
$$\begin{bmatrix} 0.4 & -0.1 & 0.1 \\ 0.2 & 0.6 & 0.0 \\ -0.5 & -0.3 & 0.9 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\frac{\left(\begin{bmatrix} 0.4 \\ 0.2 \\ -0.5 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.6 \\ -0.3 \end{bmatrix}\right)}{2} = \begin{bmatrix} 0.15 \\ 0.4 \\ -0.4 \end{bmatrix}$$

Projection Layer는 One-Hot Vector의 특성상, LUT(Look-Up Table)의 형태로 구현된다.

# Output Layer

$$\boldsymbol{y} = \text{softmax}(\boldsymbol{W}'\boldsymbol{v})$$




CBOW

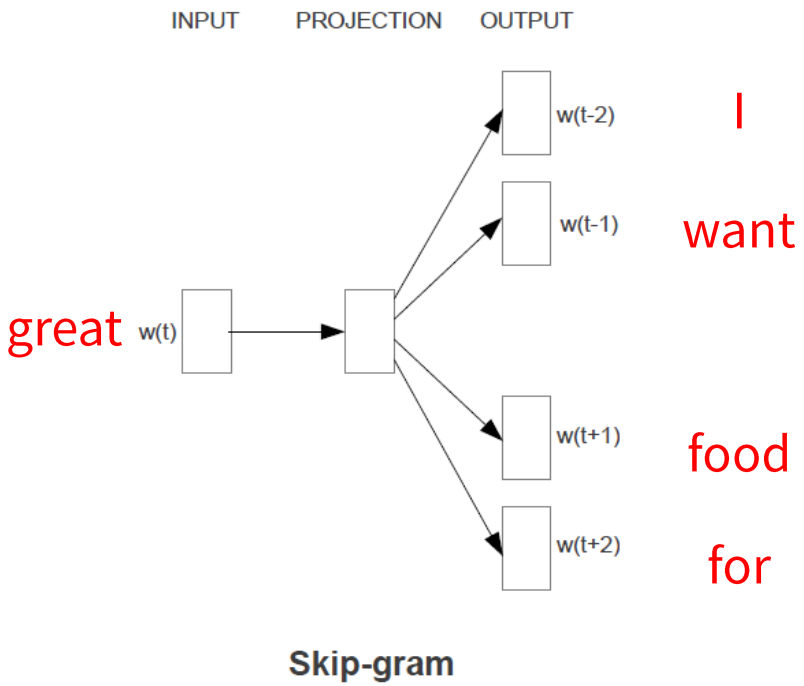
$$\begin{bmatrix} 0.5 & 0.2 & 0.4 \\ -0.5 & 0.4 & 0.2 \\ 0.1 & 0.0 & 0.7 \end{bmatrix} \begin{bmatrix} 0.15 \\ 0.4 \\ -0.4 \end{bmatrix} = \begin{bmatrix} -0.005 \\ 0.005 \\ -0.265 \end{bmatrix}$$

$$\boldsymbol{y} = \text{softmax}\left(\begin{bmatrix} -0.005 \\ 0.005 \\ -0.265 \end{bmatrix}\right) = \begin{bmatrix} 0.360 \\ 0.363 \\ 0.28 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Cross Entropy

Loss Function은 정답인 One-Hot Vector와 Cross-Entropy Loss를 이용한다.

# Skip-Gram




Skip-gram

Skip-Gram 모델은 CBOW와 반대로, Window 중앙에서 주변 단어를 추정하는 방식이다.

일반적으로 CBOW보다 Skip-Gram 모델의 성능이 좋은 것으로 알려져 있다.