

AI Solution for AI needs customer



# Word2Vec 개념과 활용

최태균 Mindmap.ai 선임연구원  
| tgchoi03@gmail.com | 010-2004-1188

V1.0 | Released in 2018.01

Document by Mr.MIND AI Consulting

Technology for network Appliance \_ Understand User,  
Create Experience **AI Consulting Inc.**

1. Word Embedding이란?

2. Word2vec모델과 알고리즘

3. Word2Vec의 활용

# 1. Word Embedding이란?

Word Embedding이란?

어떻게 사람은 단어를 표현하는가?

“사람들은 의미를 통해서 단어를 표현하고자 한다.”

정의 : “의미” (웹스터 사전)

- 단어나 구에 의해 표현되는 생각
- 단어나 신호들을 통해 사람들이 표현하고자 하는 생각
- 글이나 작업을 통해 표현되는 생각

Word Embedding이란?

어떻게 컴퓨터는 단어를 표현하는가?

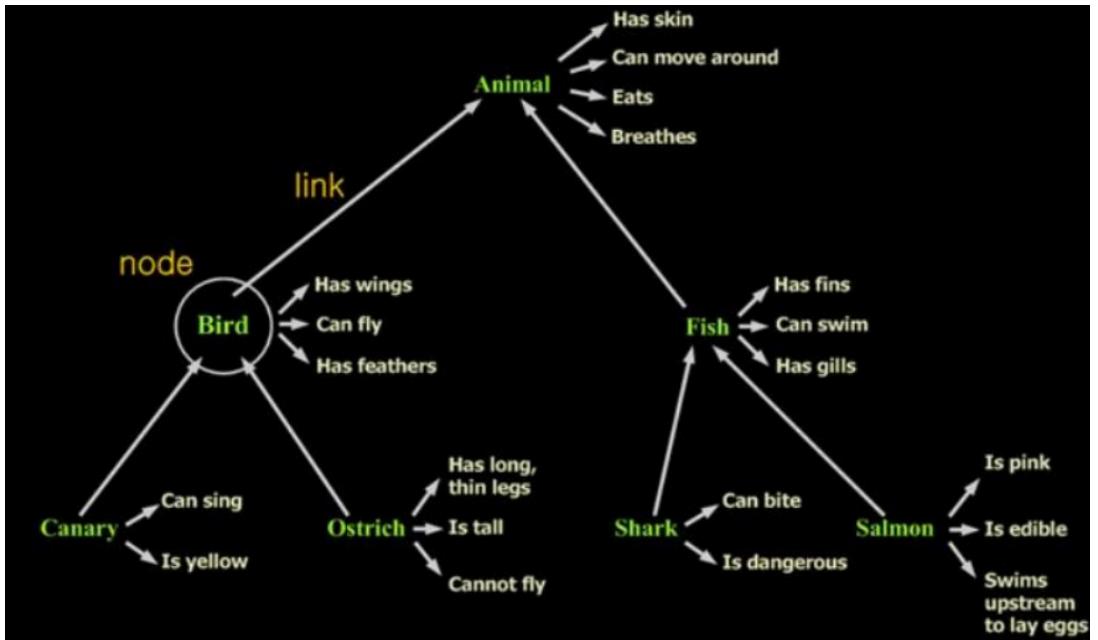
## WordNet

- 단어들에 대해 유의어 집단으로 분류하여 간략하고 일반적인 정의를 제공하고, 이러한 어휘목록 사이의 다양한 의미 관계를 기록한다.
- Is-a와 같은 상위어 관계를 가진다.

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

'panda'에 대한 유의어 집합



WordNet에 대한 그래프 표현

Word Embedding이란?

어떻게 컴퓨터는 단어를 표현하는가?

## WordNet의 한계

- 어휘의 의미 외의 내용을 포착하기 어렵다.  
i.e) “사과가 맛있다.” 와 “사과하고 싶어.”에서 ‘사과’의 의미는 동일한가?
- 사전 내의 단어에 대해서만 어휘 관계를 볼 수 있다.
- 주관적인 어휘 관계 구성이 있다.
- 상당한 노동을 요하는 작업이다.
- 단어 간의 유사도를 확인하기 어렵다.

```
from nltk.corpus import wordnet as wn
wn.synsets('nginx')
#=> []
wn.synsets('yolo') # 뜻: you only live once
#=> []
```

새로운 단어에 대해서는 어휘의 유사어를 찾기 어렵다.

Word Embedding이란?

어떻게 사람은 단어를 표현하는가?

## Word Embedding

- 한 단어를 벡터 형태로 표현하는 방법
- 대표적으로 이진(Binary) 표현법이 있다.

### 이진(Binary) 표현법

- $w_i$  은  $i$ 번 단어에 대한 벡터 표현
- 각 벡터의 크기는 모든 단어의 set인 전체 사전의 크기  $|V|$  와 같다.
- 뉴럴넷 쪽 학자들은 이렇게 표현한 벡터를 보통 one-hot vector라고 한다.
- 어떤 사람들은 이런 방식의 인코딩을 1-of-V 코딩이라고도 한다.

$$w^{\text{가}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{\text{가가(假家)}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{\text{가가대소}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \dots, w^{\text{행자}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Word Embedding이란?

어떻게 컴퓨터는 단어를 표현하는가?

## 이진표현법의 한계

- 표현할 단어의 수가 많아질 수록 벡터의 크기가 커진다.  
Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)
- 모든 단어 벡터에 대해서 직교성(orthogonality)를 가지기 때문에, 유사성을 구하기 어렵다.

$$\begin{aligned} \text{motel} & [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]^T \\ \text{hotel} & [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] = 0 \end{aligned}$$

## 기존의 방법으로 Word 간 유사도 구하려면?

- 전 슬라이드의 방법을 사용하여 term-document matrix 구축
- e.g., Document 1 = "John likes movies. Mary likes too."  
Document 2 = "John also likes football."
- 두 단어가 얼마나 자주 같은 문서에서 등장했나로 유사도를 측정

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



Co-occurrence similarity

- John & likes : 2 (documents)
- movies & also: 0 (documents)

## 2. Word2Vec모델과 알고리즘

## Distributional Similarity Based Representations

### Distributional Similarity Based Representations.

- 주변단어를 통해 해당하는 단어를 표현한다.
- Statistical NLP에서 가장 성공적인 아이디어로 꼽힌다.

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

J. R. Firth

government debt problems turning into banking crises as has happened in  
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

## Window Based Cooccurrence Matrix

어떻게 주변 단어들을 표현할 수 있는가?

- 단어들 간의 공기(cooccurrence)를 하나의 행렬로 표현한다.
- Window Size를 정하여 단어들 간의 공기를 본다.
  - 일반적으로 5~10 사이의 window size를 정한다.
  - 지정한 단어를 기준으로 symmetric하게 구성을 한다.

## Window Based Cooccurrence Matrix

### Window Based Cooccurrence Matrix

- 다음의 3개의 문장을 통해 단어들의 관계에 대한 행렬을 만든다.
  - I like deep learning
  - I like NLP
  - I enjoy flying
- Window length는 1이다. (window size는 3이다.)

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

Window Based Cooccurrence Matrix

## Window Based Cooccurrence Matrix

### Cooccurrence vector에 대한 한계

- 단어 수가 커짐에 따라 cooccurrence matrix의 크기가 커진다.
- Cooccurrence vector 특성에 대한 Sparsity가 커진다.
- 따라서, vector에 대한 차원축소를 하여 dense vector로 만든다.

## Window Based Cooccurrence Matrix

### 문제 해결 방법 1

- Singular Vector Decomposition (SVD) on Cooccurrence Matrix

$$\begin{array}{c}
 \begin{matrix} m \\ n \end{matrix} \\
 X
 \end{matrix}
 = \begin{matrix} r \\
 n \begin{matrix} U_1 U_2 U_3 \dots \end{matrix} \\
 \end{matrix}
 \begin{matrix} r \\
 r \begin{matrix} S_1 S_2 S_3 \dots 0 \\ 0 \end{matrix} \\
 S
 \end{matrix}
 \begin{matrix} m \\
 r \begin{matrix} V_1 V_2 V_3 \dots \\ \vdots \end{matrix} \\
 V^T
 \end{matrix}$$
  

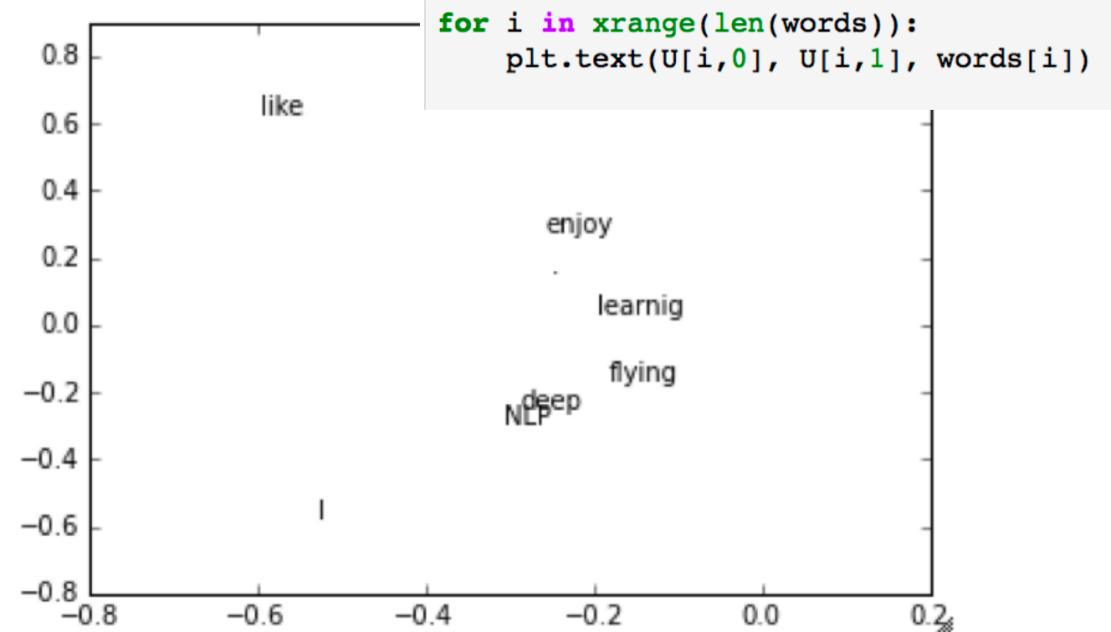
$$\begin{matrix} m \\ n \end{matrix} \\
 \hat{X}
 = \begin{matrix} k \\
 n \begin{matrix} \hat{U}_1 \hat{U}_2 \hat{U}_3 \dots \end{matrix} \\
 \end{matrix}
 \begin{matrix} k \\
 k \begin{matrix} \hat{S}_1 \hat{S}_2 \hat{S}_3 \dots 0 \\ 0 \end{matrix} \\
 \hat{S}
 \end{matrix}
 \begin{matrix} m \\
 k \begin{matrix} \hat{V}_1 \hat{V}_2 \hat{V}_3 \dots \\ \vdots \end{matrix} \\
 \hat{V}^T
 \end{matrix}$$

## Window Based Cooccurrence Matrix

### 문제 해결 방법 1

- Singular Vector Decomposition (SVD) on Cooccurrence Matrix

```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep", "learnig", "NLP", "flying", ".."]
X = np.array([[0, 2, 1, 0, 0, 0, 0, 0],
              [2, 0, 0, 1, 0, 1, 0, 0],
              [1, 0, 0, 0, 0, 0, 1, 0],
              [0, 1, 0, 0, 1, 0, 0, 0],
              [0, 0, 0, 1, 0, 0, 0, 1],
              [0, 1, 0, 0, 0, 0, 0, 1],
              [0, 0, 1, 0, 0, 0, 0, 1],
              [0, 0, 0, 0, 1, 1, 1, 0]])
U, s, Vh = la.svd(X, full_matrices=False)
```



위의 Cooccurrence Matrix 예시를 활용하여 SVD 분해를 한 후,  
가장 큰 두 singular value에 해당하는 성분벡터를 찾아 그래프를 그린 것이다.

## Window Based Cooccurrence Matrix

### SVD의 문제점

- 단어나 문서의 수가 큰 경우 연산비용이 상당히 크다.
- 한 번 만든 Word Embedding Vector에 대해서 새로운 단어나 문서에 대한 정보를 더 추가할 수가 없다.

## Word2Vec모델

### Word2Vec의 아이디어

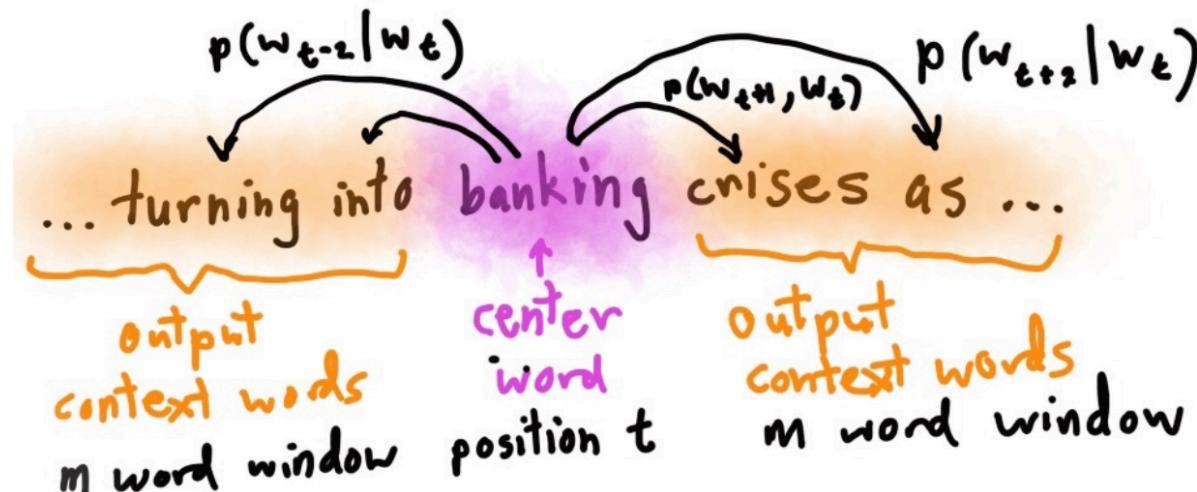
- 앞선 cooccurrence matrix를 활용하지 않고, 단어를 통해 주변 단어들을 예측하는 모델을 만든다.
- SVD 방식에 비해 연산이 빠르고, 새로운 어휘나 문서를 추가할 수 있다.
- CBOW 방식 : Window내 주변 단어를 중심으로 단어를 예측
- Skip-gram 방식 : 중심단어를 기준으로 window내 주변 단어를 예측



## Word2Vec모델 알고리즘

### Skip-gram 모델

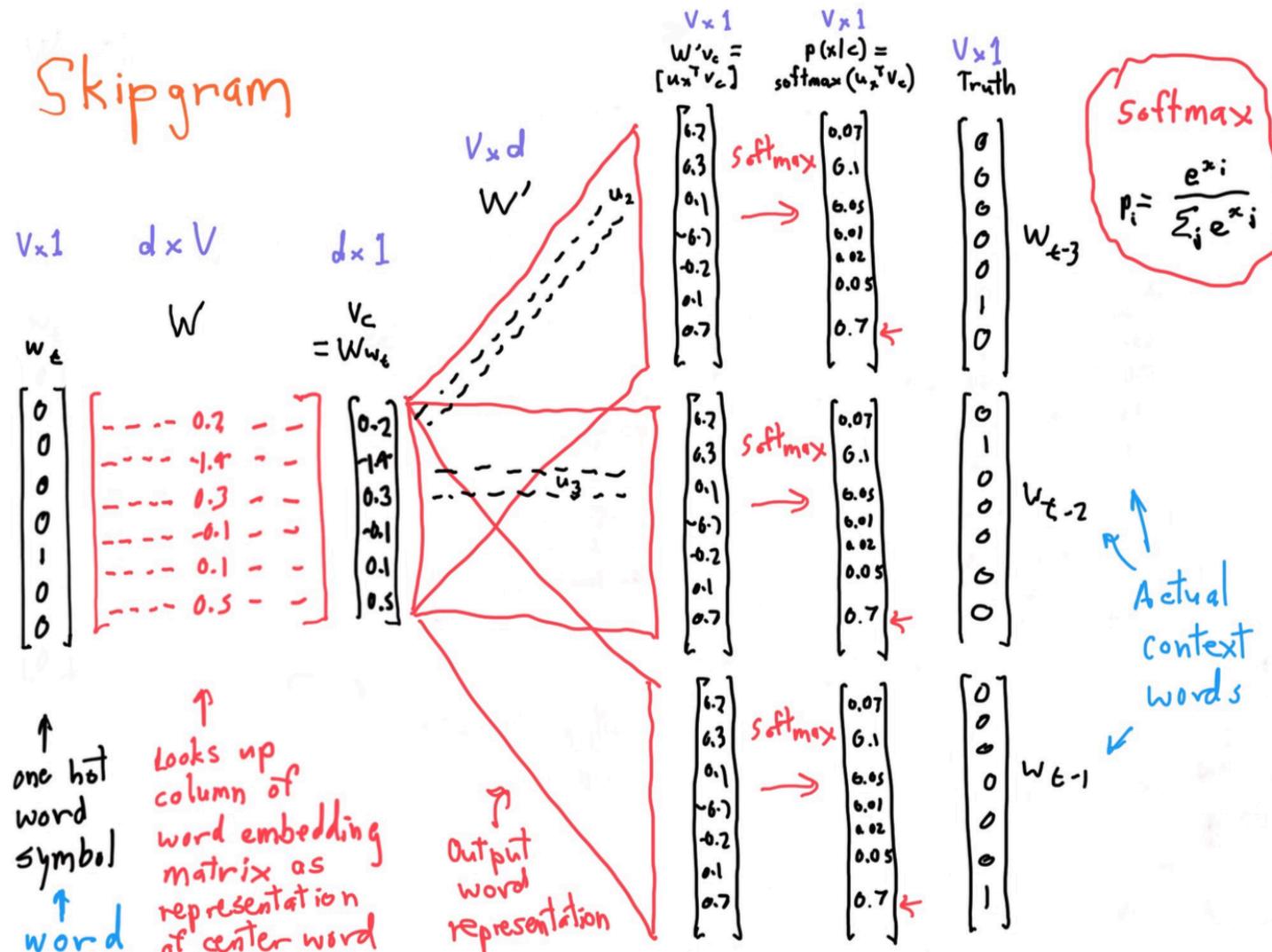
- Window의 중심 단어를 모델의 input으로 넣어 output이 주변 단어들로 예측할 수 있도록 하는 모델



Skip-gram 예측

## Word2Vec 모델 알고리즘

## Skip-gram 모델



## Word embedding matrix

- Initialize most word vectors of future models with our “pre-trained” embedding matrix  $L \in \mathbb{R}^{n \times |V|}$

$$L = \begin{bmatrix} \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots \end{bmatrix}_n^{|V|}$$

aardvark    a       at    ...

- Also called a look-up table
  - Conceptually you get a word’s vector by left multiplying a one-hot vector  $e$  (of length  $|V|$ ) by  $L$ :  $x = Le$

**Word2Vec모델 알고리즘****Skip-gram모델의 목적함수**

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

목적함수 : 중심단어에서 주변단어를 예측하는 Log Probability의 합

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

Softmax 함수

## Skip-gram의 문제점

### Skip-gram의 학습 한계

- 어휘의 총 개수가 많을 경우 학습시간이 상당히 느린다.  
i.e) Window Size가 5이고 총 어휘가 10만개인 경우, 학습 시 10만개의 classification을 총 4번 시행해야 한다.
- 특히 Softmax 연산을 하는 과정에서 상당한 곱셈 연산을 하기 때문에 연산비용이 상당하다.

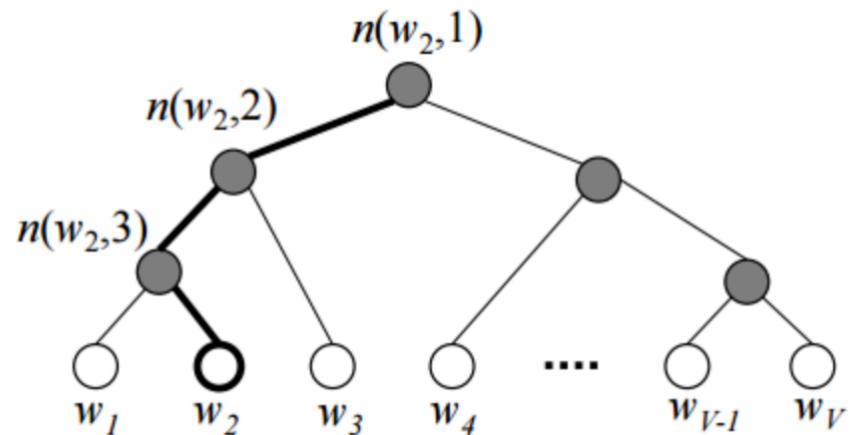
$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

Softmax 함수

## Skip-gram의 문제 해결

### Hierarchical Softmax

- 각 단어들을 leaves로 가지는 binary tree를 생성
- 해당하는 단어의 확률을 계산할 때 root에서 해당 leaf로 가는 path에 따라 확률을 곱한다.
- 매 분기의 각 양쪽의 path에 대한 확률의 합은 1이다.
- 정답이 되는 확률에 대한 연산을  $n$  번에서  $\log n$  번으로 줄일 수 있다.



$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma \left( [n(w, j+1) = \text{ch}(n(w, j))] \cdot \mathbf{v}'_{n(w,j)}^T \mathbf{h} \right)$$

## Negative Sampling

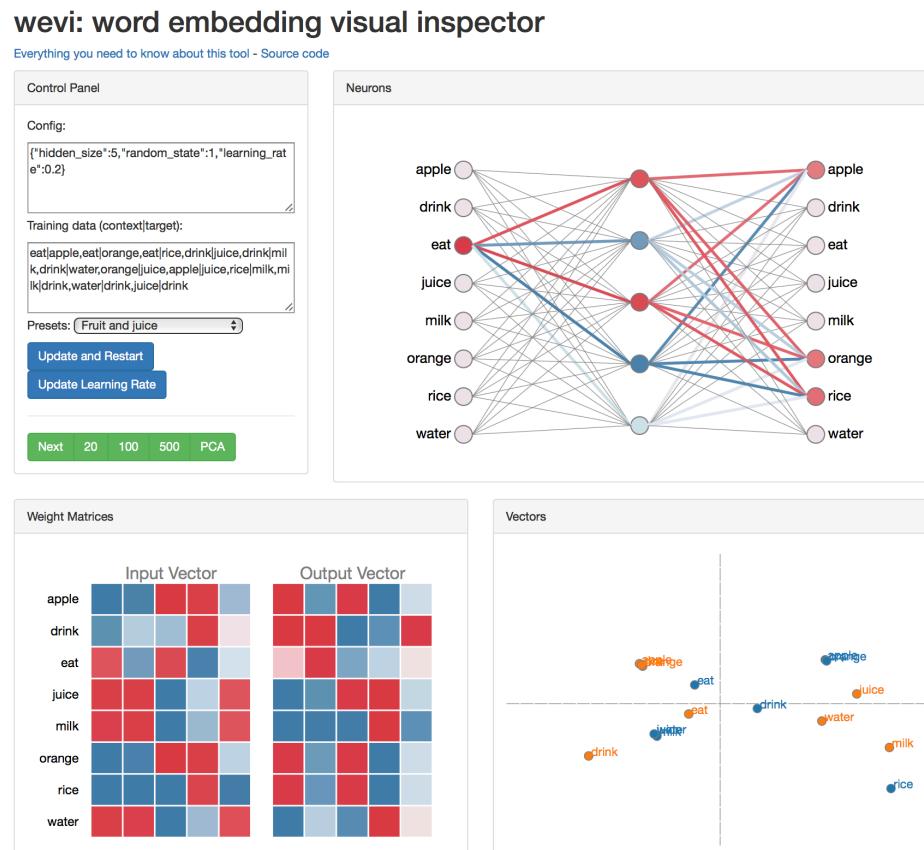
- Softmax 연산하는 과정에서 분모부분에서 일부 단어만 샘플링하여 계산하도록 함.
- 실제 target으로 사용하는 단어의 경우에 반드시 연산을 하게되는데 이를 'positive sample'이라 하고, 나머지를 'negative sample'라고 함.
- Negative Sampling을 하는 과정은 'Noise Distribution'이라 정의 하고, 그 분포에서  $P_n(w)$ 를 이용하여 단어들을 일정 갯수 뽑아서 사용한다. 실험적으로 'Unigram Distribution의 ¾ 제곱'으로 설정.

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

## Word2Vec Visualization

# Word Embedding Visual Inspector

- <https://ronxin.github.io/wevi/>



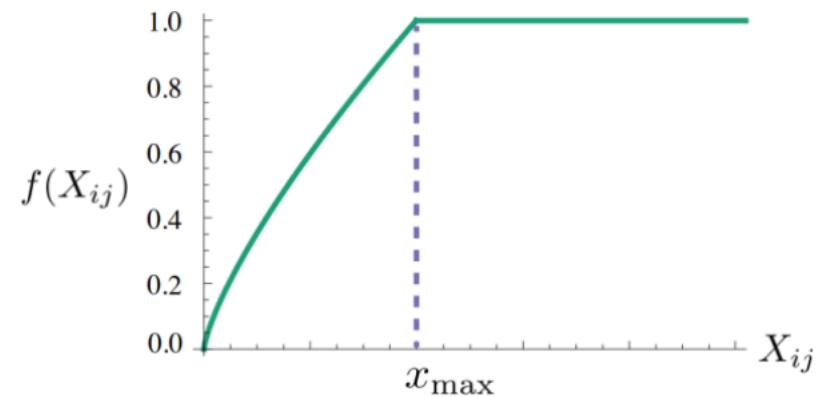
## Glove모델

### Glove모델

- Cooccurrence Matrix  $P$ 의 정보를 가지고 동시에 등장하는 주변단어의 확률을 최대한 보존하고자 하는 방법.
- 행렬  $P$ 를 분해하여 성분분석을 하는 것이 아니라, 목적함수를 통해 임의의 단어 백터  $u_i$ 와  $u_j$ 에 대한 유사도를 최대한  $\log(P_{ij})$ 에 가깝도록 한다.

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

목적함수



## Glove모델

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

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

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

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

## Glove모델

### Glove모델 장점

- 빠른 학습이 가능하다.
- 작은 corpora에서도 효과적이다.
- 많은 NLP 모델에서 Word Vector로 활용하고 있다.

**GloVe: Global Vectors for Word Representation**

Jeffrey Pennington, Richard Socher, Christopher D. Manning

**Introduction**

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

**Getting started (Code download)**

- Download the [code](#) (licensed under the [Apache License, Version 2.0](#))
- Unpack the files: `unzip GloVe-1.2.zip`
- Compile the source: `cd GloVe-1.2 && make`
- Run the demo script: `./demo.sh`
- Consult the included README for further usage details, or ask a [question](#)
- The code is also available [on GitHub](#)

**Download pre-trained word vectors**

- Pre-trained word vectors. This data is made available under the [Public Domain Dedication and License](#) v1.0 whose full text can be found at: <http://www.opendatacommons.org/licenses/pddl/1.0/>.
  - Wikipedia 2014 + Gigaword 5 (68 tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): [glove.6B.zip](#)
  - Common Crawl (42B tokens, 19M vocab, uncased, 300d vectors, 1.75 GB download): [glove.42B.300d.zip](#)
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): [glove.840B.300d.zip](#)
  - Ruby (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): [glove.twitter.27B.zip](#)
- Ruby [script](#) for preprocessing Twitter data

**Citing GloVe**

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*. [\[pdf\]](#) [\[bib\]](#)

**Highlights**

**1. Nearest neighbors**

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

o. <i>frog</i>	1. <i>frogs</i>	2. <i>toad</i>	3. <i>litoria</i>
4. <i>leptodactylidae</i>	5. <i>rana</i>	6. <i>lizard</i>	7. <i>eleutherodactylus</i>






3. *litoria*      4. *leptodactylidae*      5. *rana*      7. *eleutherodactylus*

Glove 홈페이지

## Word2Vec 결과

### Word2Vec의 특징

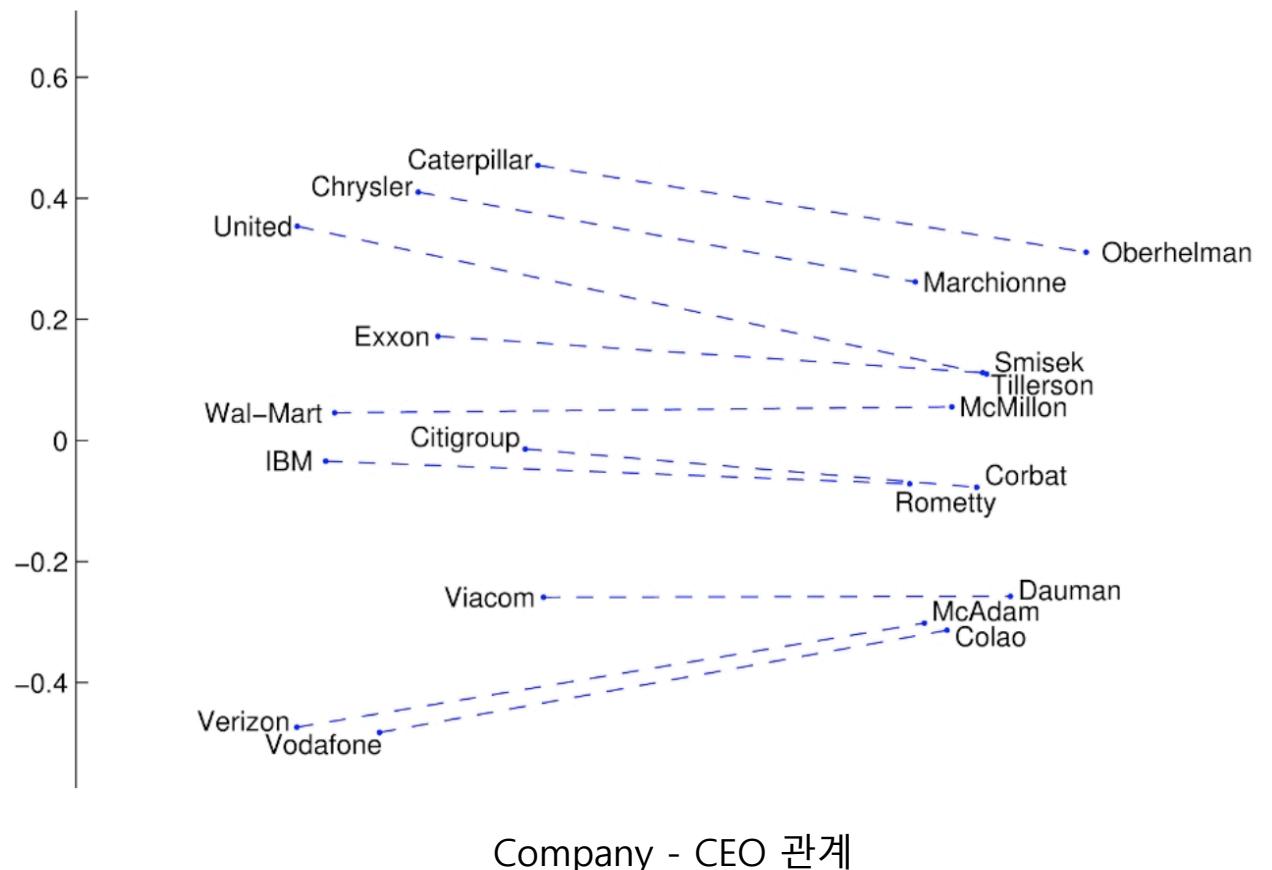
- 단어 간의 관계를 보다 명확히 찾는다.
- Analogies testing 사례에서 단순히 단어 벡터들 간의 차를 통해서 단어 간의 관계를 찾는 것을 볼 수 있었다.
  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$  (Syntactic 관계)
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$  (Semantic 관계)



(Mikolov et al., NAACL HLT, 2013)

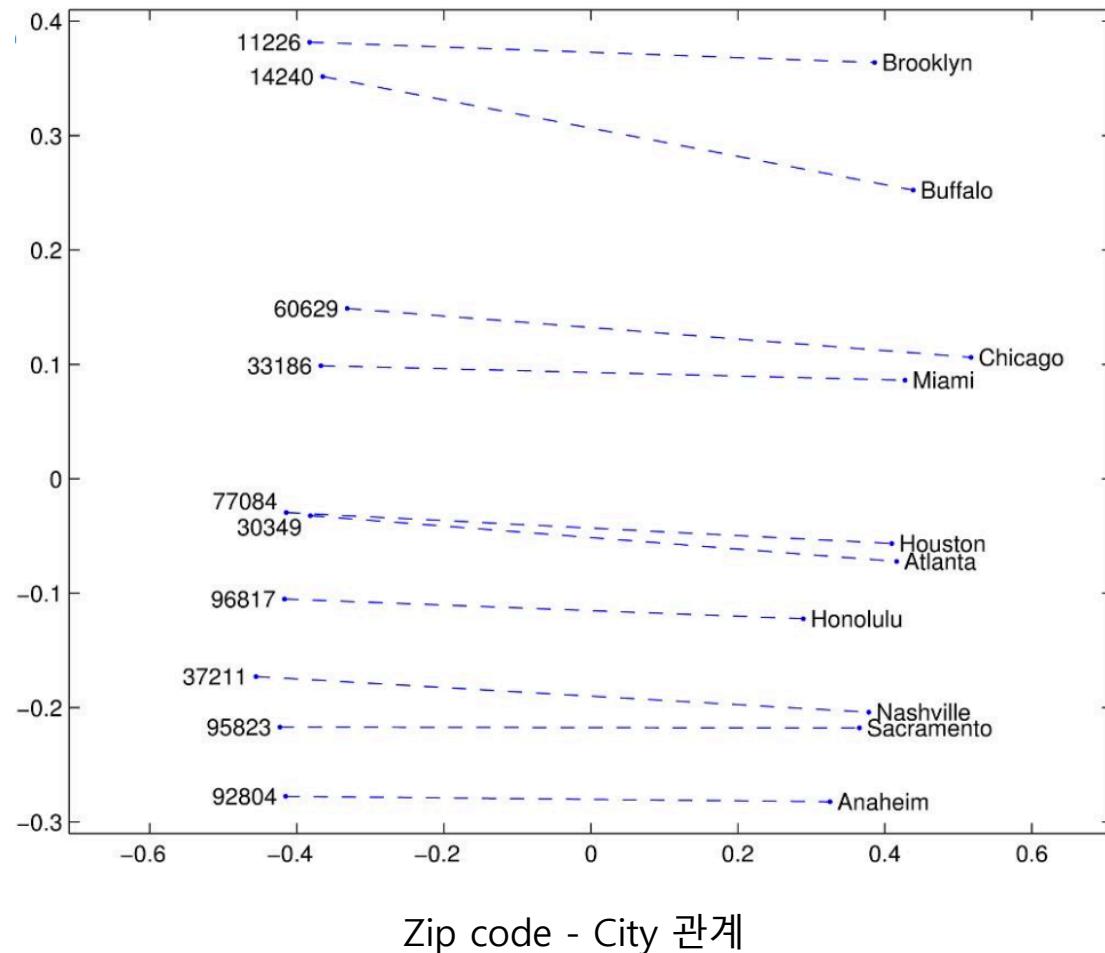
## Word2Vec 결과

### Word2Vec의 특징



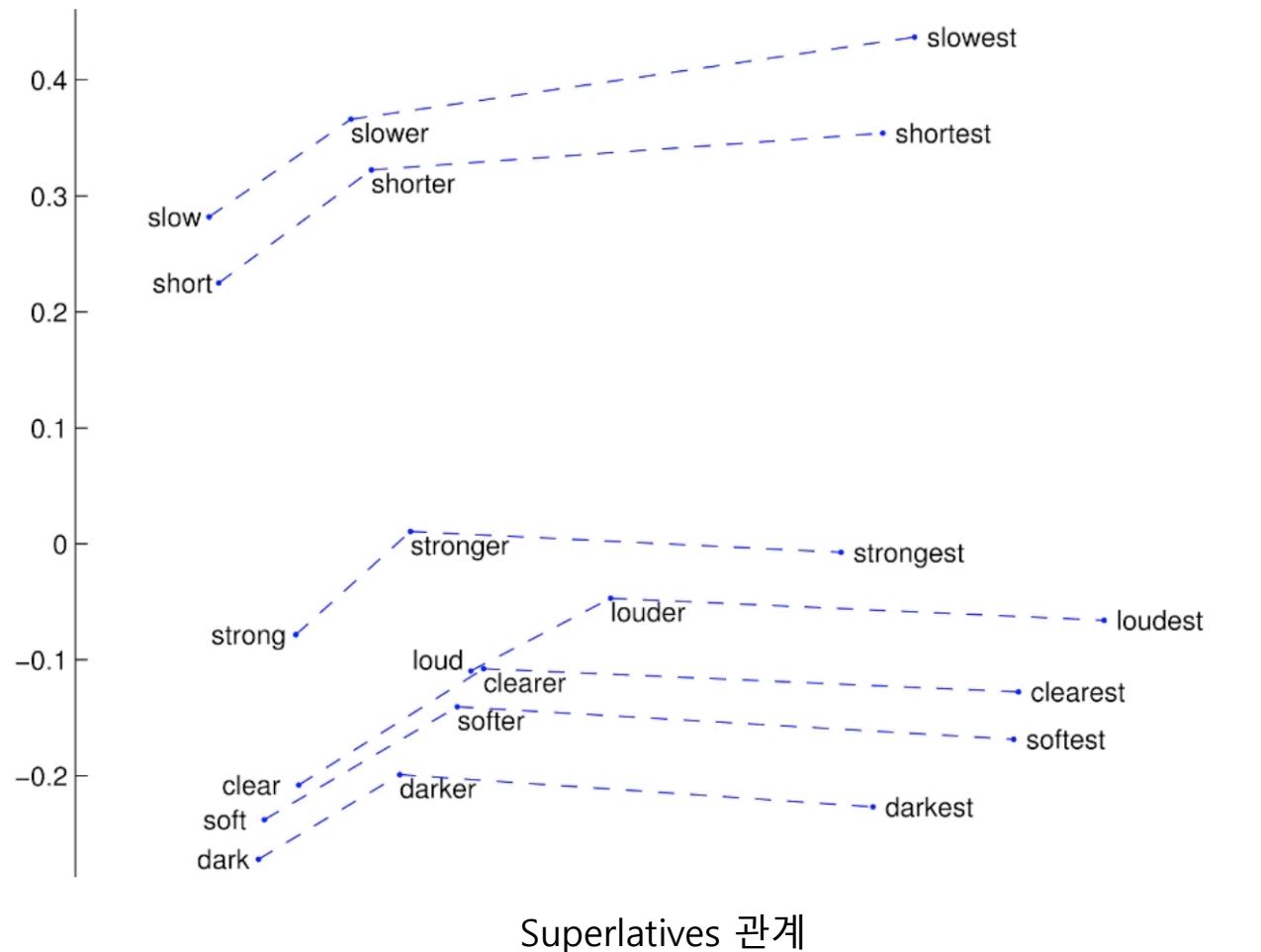
## Word2Vec 결과

### Word2Vec의 특징



## Word2Vec 결과

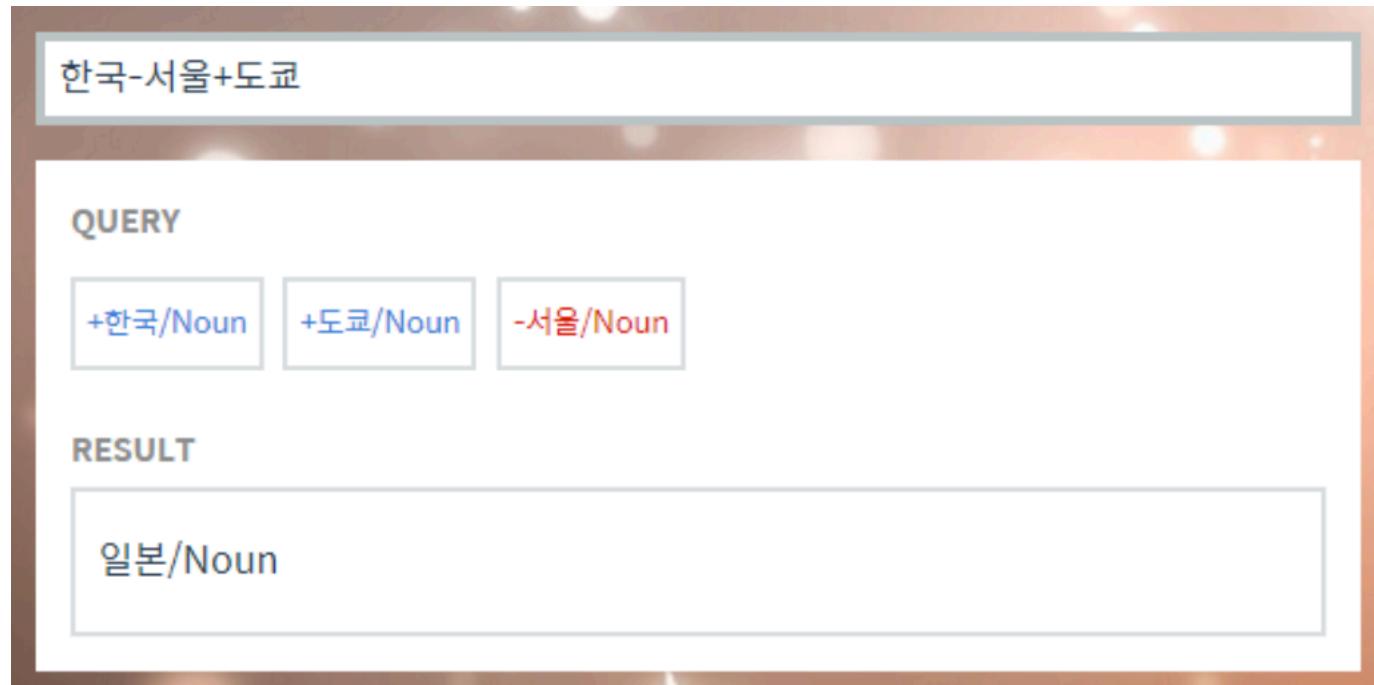
### Word2Vec의 특징



## Word2Vec 결과

### Word2Vec 벡터 연산 실습

한글 Word2Vec 벡터 연산 => <http://w.elnn.kr/search/>



## Word2Vec 결과

### Word2Vec 벡터 연산 실습 예시

데모	<a href="http://w.elnn.kr/">http://w.elnn.kr/</a>	
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

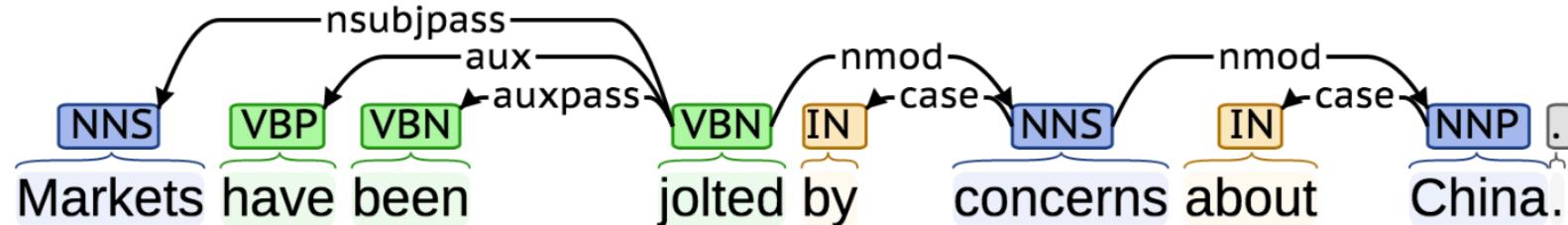
컴퓨터-기계+인간	운영체제/Noun	일반인/Noun
게임+공부	프로그래밍/Noun	덕질/Noun
박보영-배우+가수	애프터스쿨/Noun	허각/Noun
밥+했는지	끓였/Verb	저녁밥/Noun
사랑+이별	그리움/Noun	추억/Noun
삼성-한화	노트북/Noun	후지필름/Noun
소녀시대-소녀+아줌마	아이유/Noun	에이핑크/Noun
수학-증명	경영학/Noun	이산수학/Noun
스파게티-소시지+김치	칼국수/Noun	비빔국수/Noun
아버지-남자+여자	어머니/Noun	어머니/Noun
아이유-노래+연기	송중기/Noun	송중기/Noun
안드로이드-자유	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	-

## 4. Word2Vec의 활용

## 활용사례

### Part of Speech (POS) tag Parser

- 주변 어휘정보를 통해 품사를 판별하고 어휘간의 의존관계를 확인한다.



**Softmax layer:**

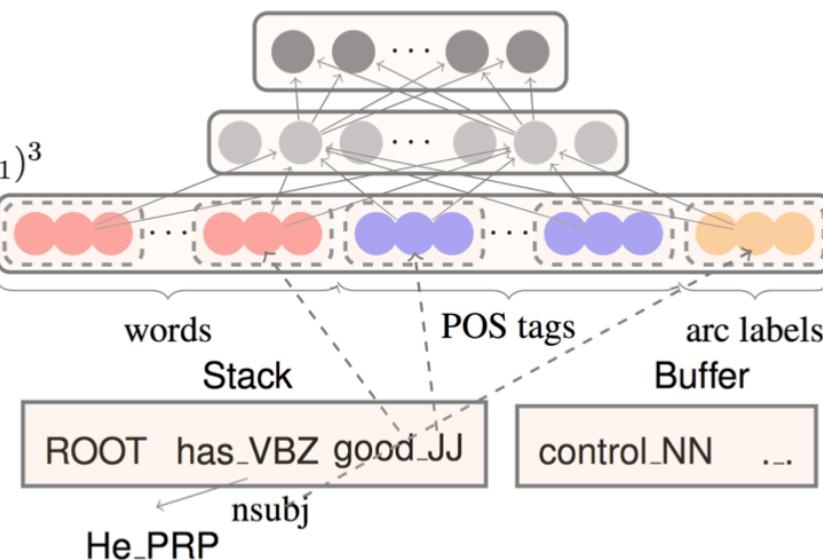
$$p = \text{softmax}(W_2 h)$$

**Hidden layer:**

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

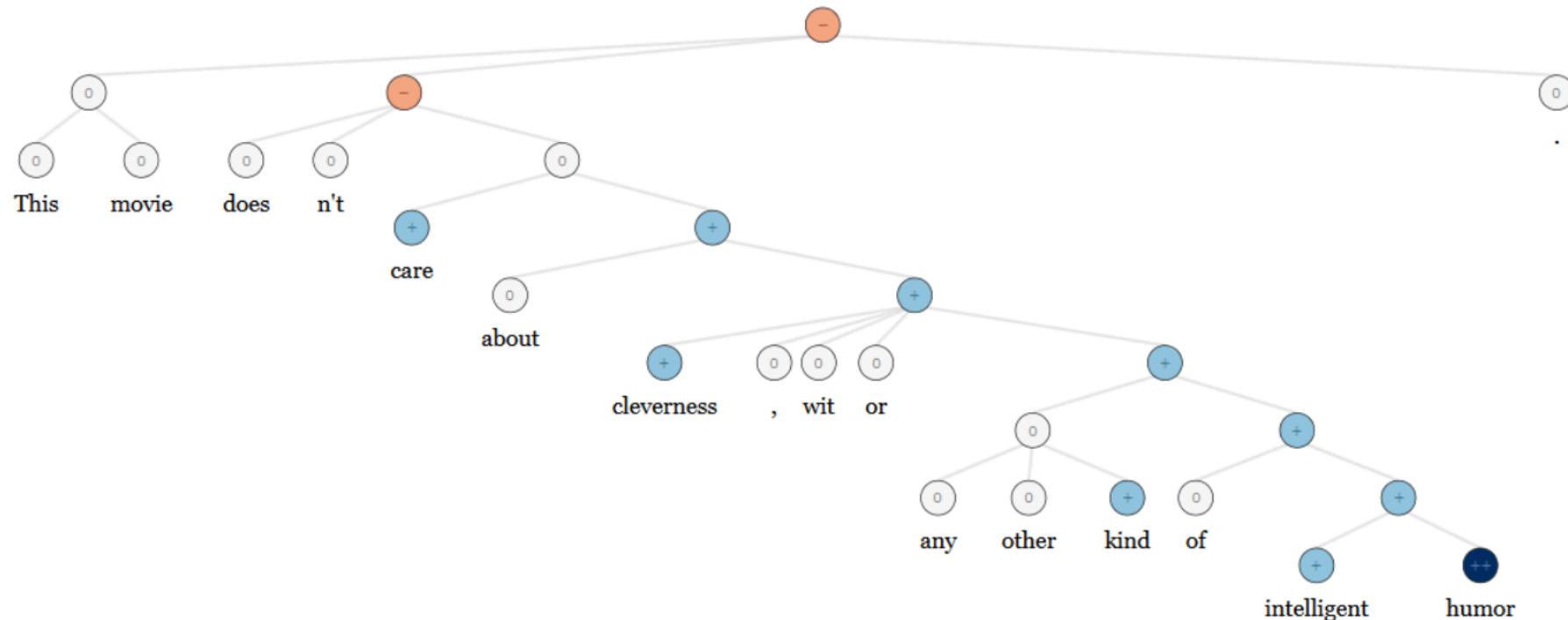
**Input layer:**  $[x^w, x^t, x^l]$

**Configuration**



## Sentiment Analysis

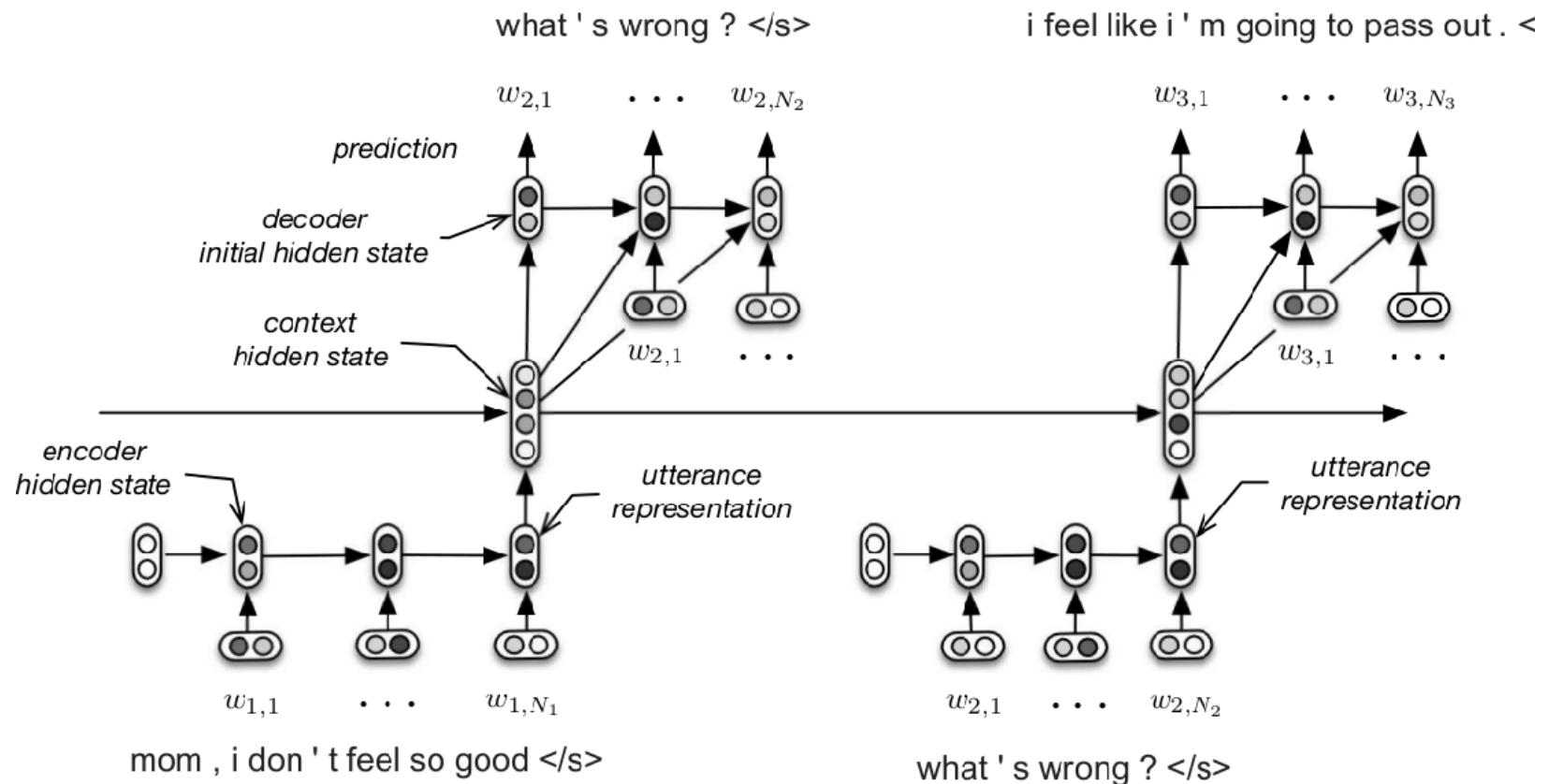
- 글에 대한 내용을 긍정인지 부정인지 여부에 대해 판별한다.



## 활용사례

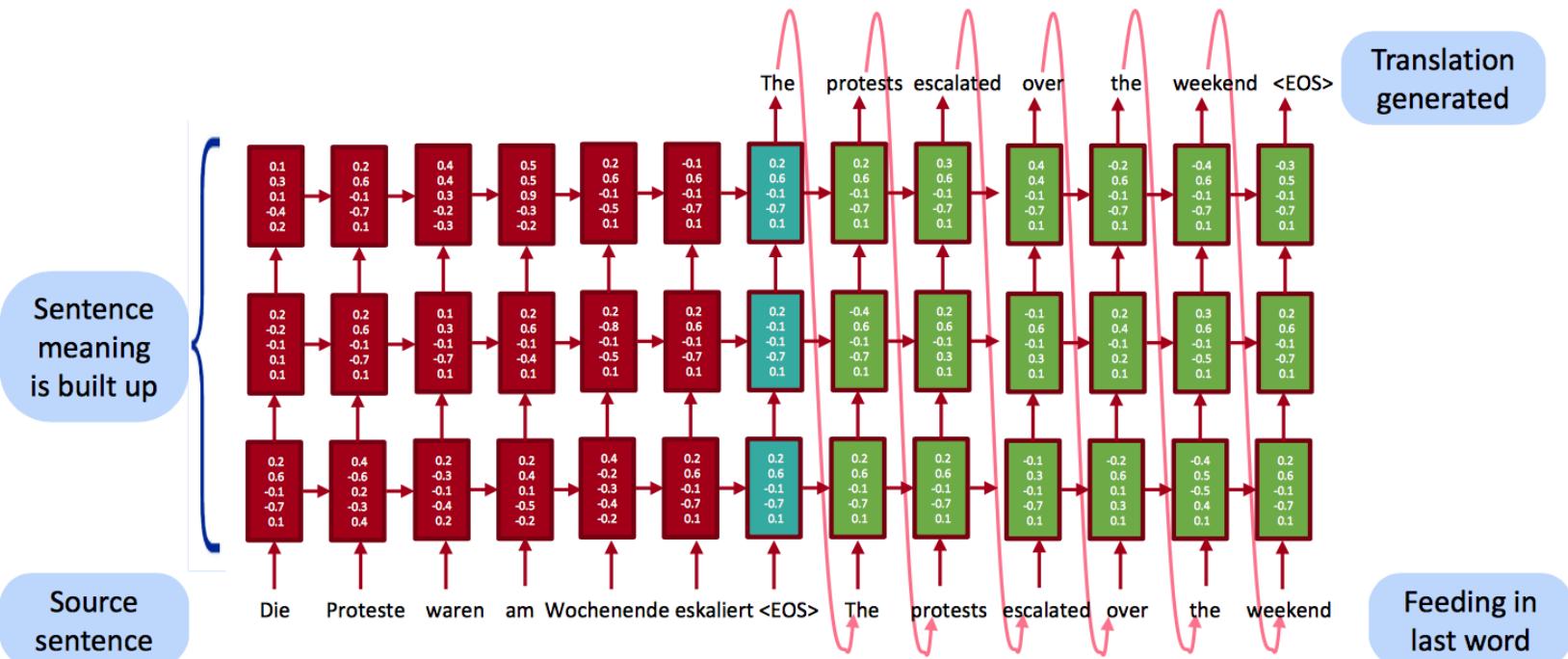
# Dialogue agents & Response Generation

- 대화 내용을 벡터 정보로 저장하여 그에 대한 반응을 생성한다.



## Neural Machine Translation

- 번역하고자 하는 내용을 Encoder에 넣어 Decoder를 통해 지정된 언어에 대한 글로 나타낸다.



## 활용사례

# Image Captioning

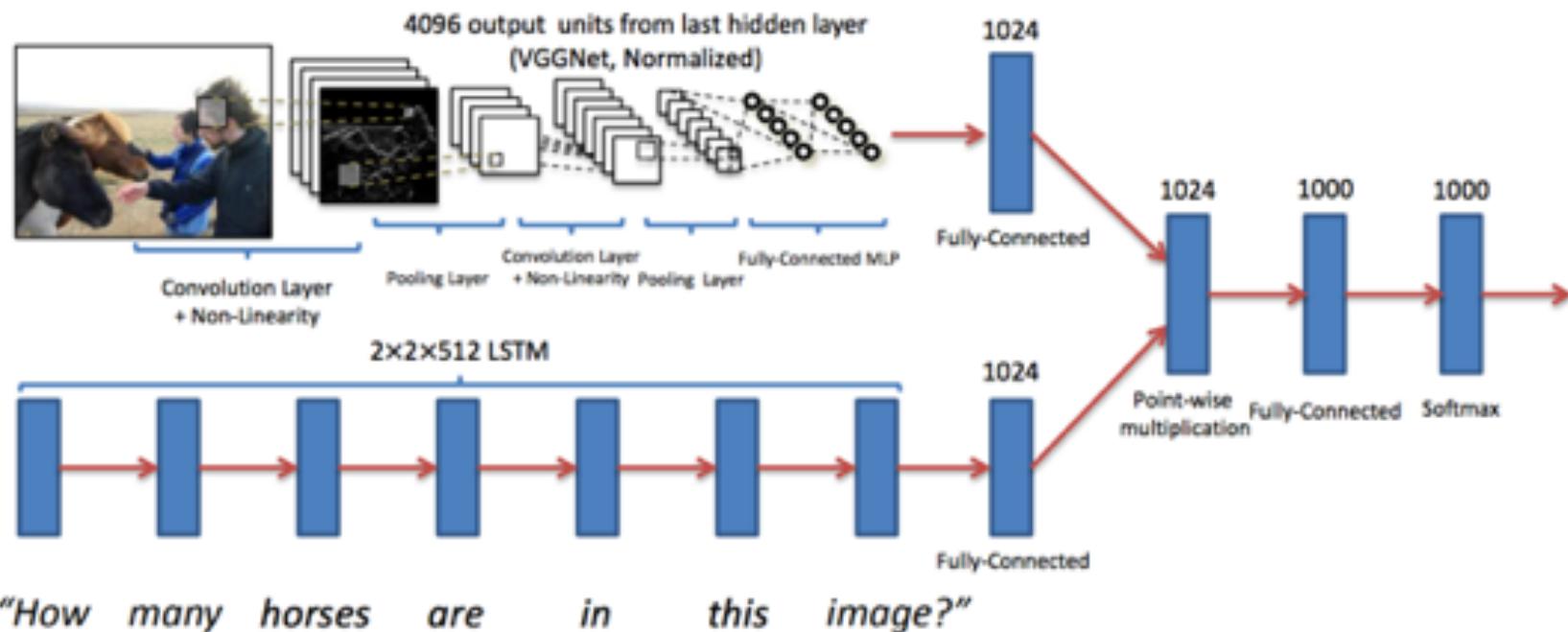
- 이미지 정보를 통해 문장을 생성한다.

Describes without errors	Describes with minor errors	Somewhat related to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.

## 활용사례

### Visual Question Answering

- 이미지에 대한 정보에 대해 텍스트 질문을 통해 답을 한다.



## 활용사례

# Machine Comprehension

- 텍스트 내용에 대해 이해하고 질문에 알맞는 정답을 텍스트 또는 정답 선택지에서 찾도록 한다.

**Super Bowl 50** was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion **Denver Broncos** defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third **Super Bowl** title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the **50th Super Bowl**, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each **Super Bowl** game with Roman numerals (under which the game would have been known as "**Super Bowl L**"), so that the logo could prominently feature the Arabic numerals **50**.

**Who won Super Bowl 50?**

Ground Truth Answers: **Denver Broncos** **Denver Broncos** **Denver Broncos**

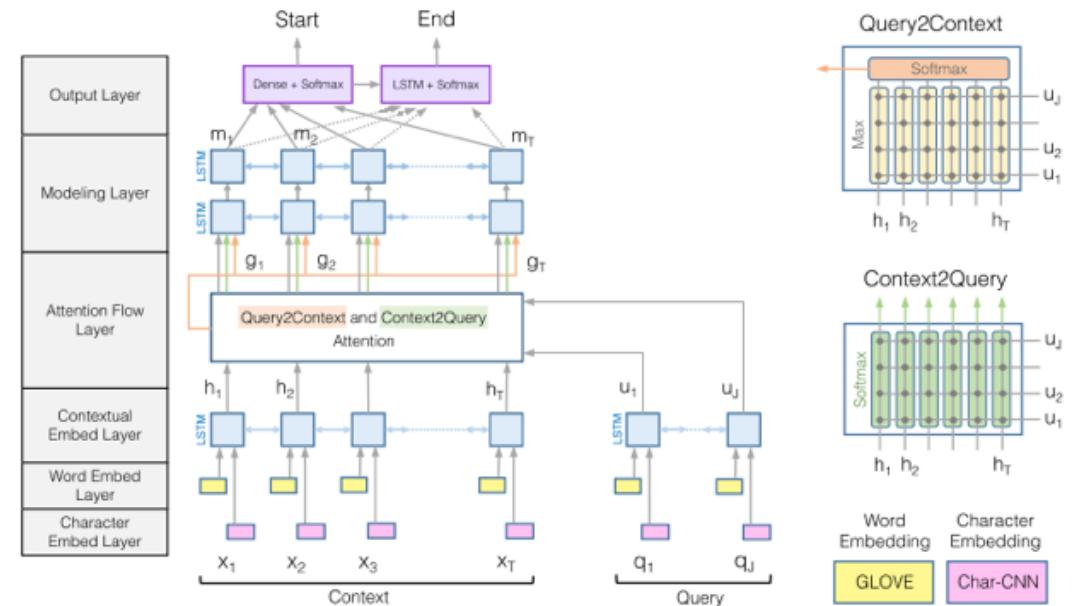


Figure 1: BiDirectional Attention Flow Model (*best viewed in color*)

## References

- Stanford CS224n Lectures  
<http://web.stanford.edu/class/cs224n/syllabus.html>
- "Word2Vec 관련 이론 정리" by 김범수님  
<https://shuuki4.wordpress.com/2016/01/27/word2vec-관련-이론-정리/>
- "Glove를 이해해보자" by 이지창님  
<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/09/glove/>
- 한국어와 NLTK, Gensim의 만남 by 박은정 박사님  
<https://www.lucypark.kr/docs/2015-pyconkr/#1>

# 감사합니다