

목차

- 자연어 처리 (Natural Language Processing, NLP)
 - 자연어란?
 - 다양한 자연어 처리
 - Word embedding
- 언어 모델 (Language Model, LM)
 - Markov 확률 모델
 - RNN (Recurrent Neural Network) 모델
 - Attention 모델
 - Self-attention 모델
 - Transformer 모델
- BERT 소개
 - WordPiece tokenizing
 - BERT 모델
 - BERT의 적용 실험
- 한국어 BERT
 - ETRI KorBERT
- BERT 실습

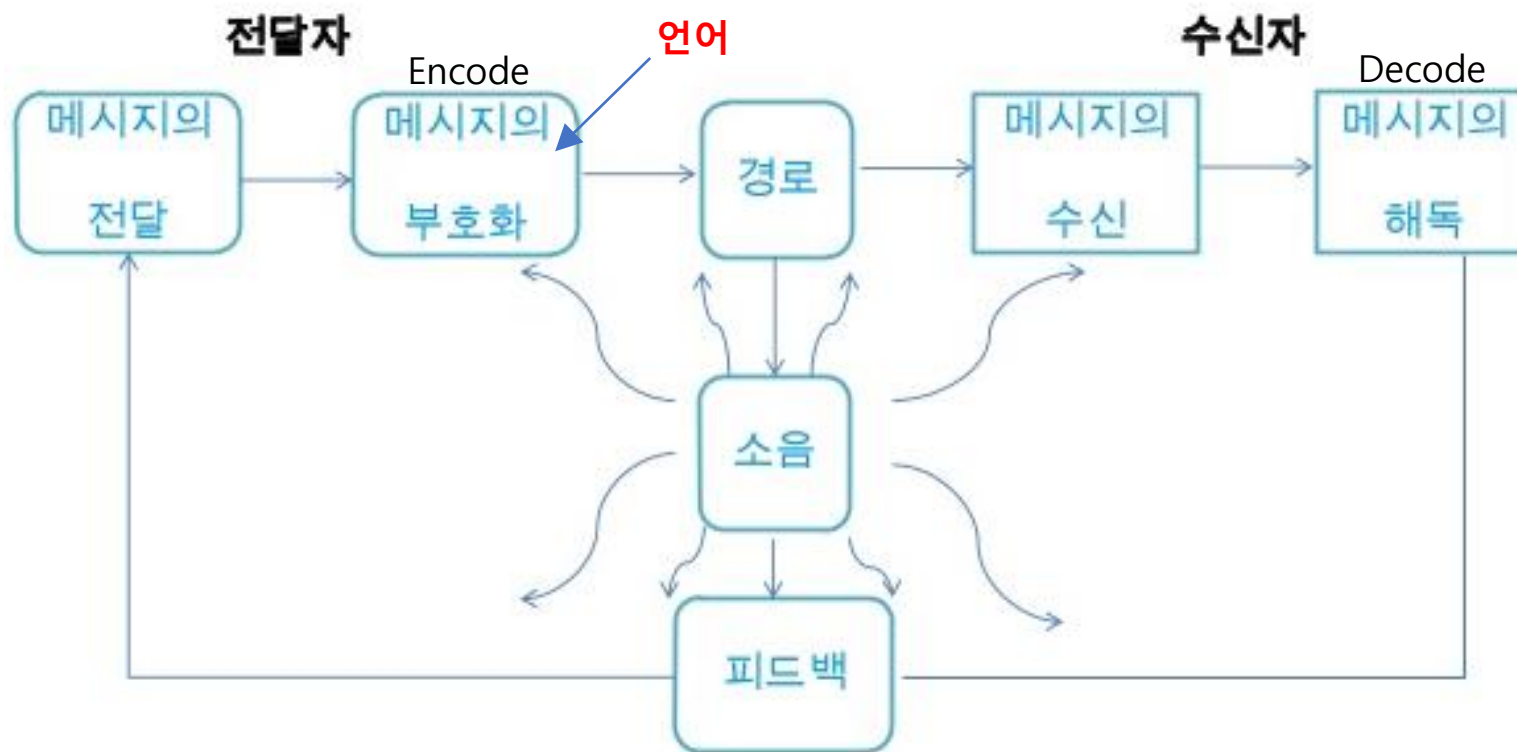
자연어 처리

- 자연어의 개념과, 자연어를 수학적으로 표현하는 방법에 대해 알아보겠습니다

자연어란?

언어¹ 言語 🗣️ ★ +

명사 생각, 느낌 따위를 나타내거나 전달하는 데에 쓰는 음성, 문자 따위의 수단. 또는 그 음성이나 문자 따위의 사회 관습적인 체계.



자연어란?

자연 언어 自然言語 +

언어 일반 사회에서 **자연**히 발생하여 쓰이는 언어.



자연언어 : 한국어, 영어, 일본어



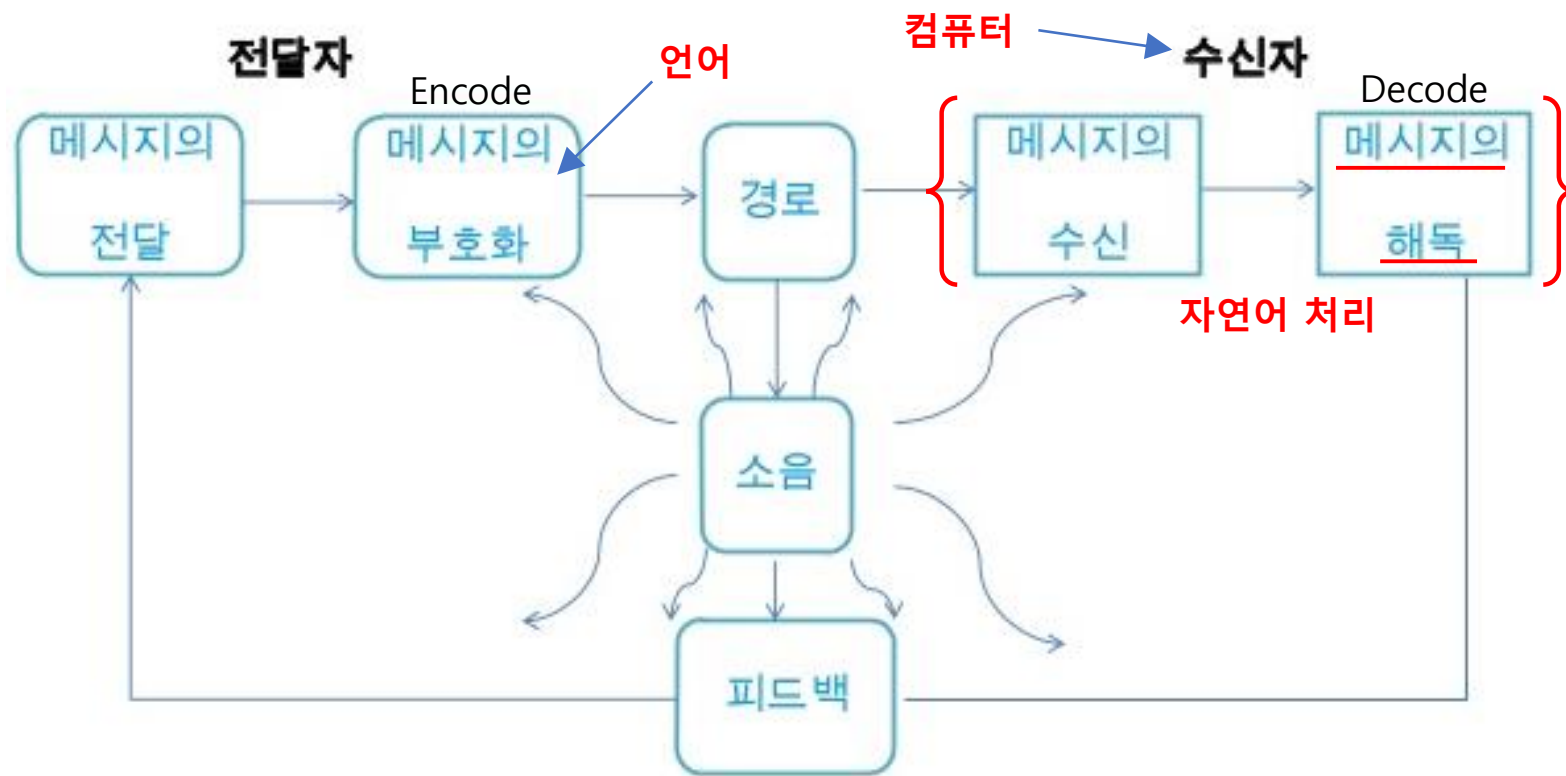
인공언어 : 프로그래밍 언어, 에스페란토어

* Chisung Song, 시나브로 배우는 자연어처리 바벨피쉬 송치성
<https://www.slideshare.net/shuraba1/ss-56479835>

자연어 처리란?

자연어 처리 自然語處理 +

정보·통신 컴퓨터를 이용하여 인간 언어의 이해, 생성 및 분석을 다루는 인공지능 기술.



01 | 자연어 처리 (Natural Language Processing, NLP)

다양한 자연어 처리 기술

- 자연어 처리란, '자연어를 컴퓨터가 해독하고 그 의미를 이해하는 기술'

Symbolic approach

- 규칙/지식 기반 접근법

```
@Override
boolean isAnswerableQuestion(String messagePattern) {
    boolean isAnswerable = false;

    if (messagePattern.matches("ChannelNm(NOW)?(PROGRAM)?WHAT")) {
        // 국회TV에서 지금 뭐해?
        isAnswerable = true;
    } else if (messagePattern.matches("ChannelNm(NOW)?WHATPROGRAM")) {
        // 국회TV에서 지금 무슨 방송해?
        isAnswerable = true;
    } else if (messagePattern.matches("ChannelNm(NOW)?PROGRAMHOW")) {
        // 국회TV에서 지금 무슨 방송해?
        isAnswerable = true;
    }
    return isAnswerable;
}
```

100 원	100 (Number) + 원(G_ExchangeRateKRW : KRW=원)
100 달러	100(Number) + 달러(G_ExchangeRateKRW : USD=달러), S_UNIT_USD_money
100 m	100(Number) + m (S_UNIT_m_length)
100 미터	100(Number) + m (S_UNIT_m_length)

Statistical approach

- 확률/통계 기반 접근법
- TF-IDF를 이용한 키워드 추출
 - TF (Term frequency): 단어가 문서에 등장한 개수
→ TF가 높을수록 중요한 단어
 - DF (Document frequency): 해당 단어가 등장한 문서의 개수
→ DF가 높을수록 중요하지 않은 단어

$$w_{x,y} = tf_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

* sujin oh, 실생활에서 접하는 빅데이터 알고리즘

https://www.slideshare.net/osujin121/ss-44186451?from_action=save

자연어 처리의 단계

• 전처리

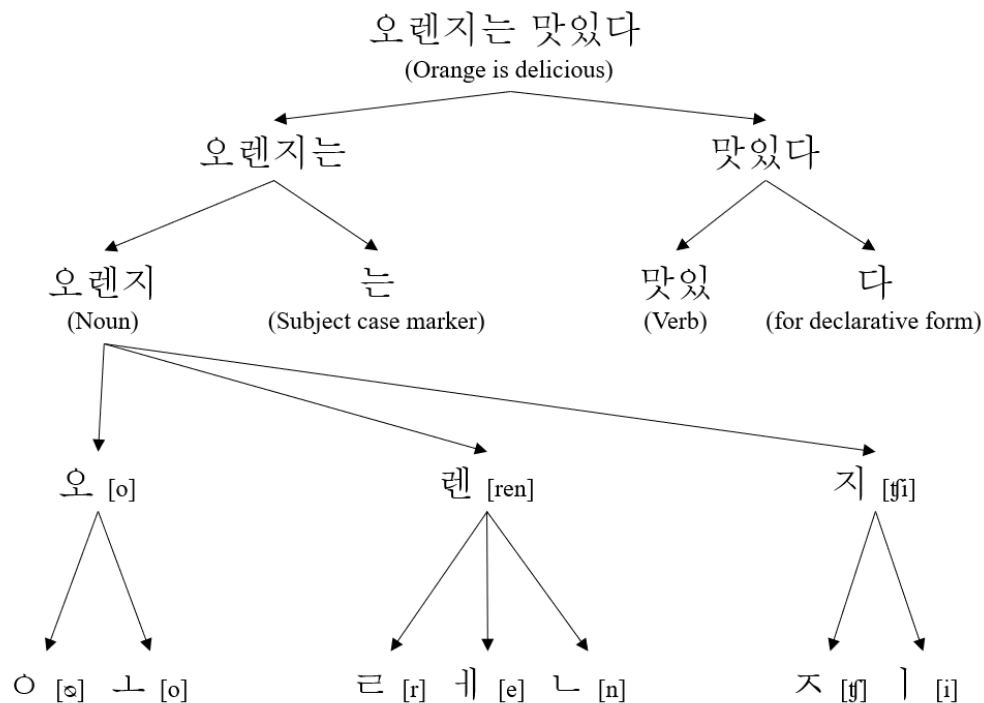
- 개행문자 제거
 - 특수문자 제거
 - 공백 제거
 - 중복 표현 제어 (ㅋㅋㅋㅋㅋ, πππππ, ...)
 - 이메일, 링크 제거
 - 제목 제거
 - 불용어 (의미가 없는 용어) 제거
 - 조사 제거
 - 띄어쓰기, 문장분리 보정
 - 사전 구축
- Tokenizing
 - Lexical analysis
 - Syntactic analysis
 - Semantic analysis

의심한그득+. + 앞으로는 사람들 많을 퇴근시간 말구
미리 가서 **사와야겠뜸** ('ù')
맥주로 터진 입, 청포도와 **고구농스틱**으로 달래봘니당
막상 먹다보니 양이 부족하진 않았는데 괜히 ...
먹을 때 많이 먹자며 배 터지기 직전까지 **남냐미:D**
식후맹 아슈크림 ♪ (>____< ♪
브라우니쿠키 먹고 출근하세요 ♡
gs편의점에서 파는 **진 - 한** 브라우니 쿠키인데 **JMTgr**



자연어 처리의 단계









- 전처리
- Tokenizing
 - 자연어를 어떤 단위로 살펴볼 것인가
 - 어절 tokenizing
 - 형태소 tokenizing
 - n -gram tokenizing
 - WordPiece tokenizing
- Lexical analysis
 - 어휘 분석
 - 형태소 분석
 - 개체명 인식
 - 상호 참조
- Syntactic analysis
 - 구문 분석
- Semantic analysis
 - 의미 분석



다양한 자연어 처리 Applications

- 문서 분류
- 문법, 오타 교정
- 정보 추출
- 음성 인식결과 보정
- 음성 합성 텍스트 보정
- 정보 검색
- 요약문 생성
- 기계 번역
- 질의 응답
- 기계 독해
- 챗봇
- 형태소 분석
- 개체명 분석
- 구문 분석
- 감성 분석
- 관계 추출
- 의도 파악

다양한 자연어 처리 Applications

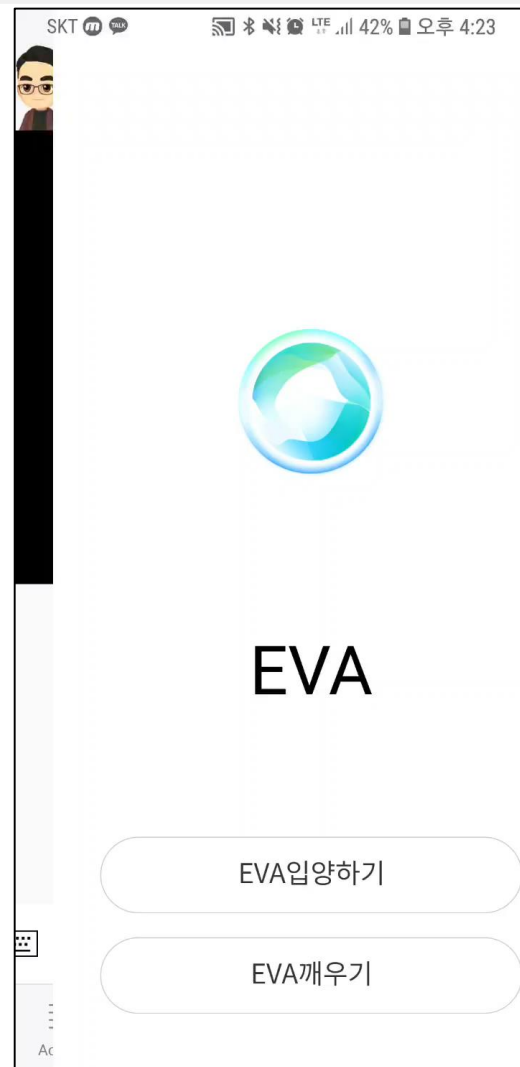
데이터	분석	언어	지식	지능	엑소브레인
					
<p>문서 필터 HTML 랩퍼 데이터셋목록 검색 데이터 조회 리소스 다운로드</p> <hr/> <p>온디맨드데이터 수집 데이터검색 데이터다운로드 데이터 업로드 데이터학습추천 데이터알림</p>	<p>트랜드(키워드) 분석 키워드 추출 연관 주제어 분석 토픽 트랜드 분석 감성분석 자동 분류 소셜 데이터(문서) 검색 오늘의 토픽 분석</p> <hr/> <p>인용구분석 관계도 분석 자동군집(v1) 자동군집(v2) 이슈감지</p>	<p>형태소분석 구문분석 개체명 인식 워드임베딩 언어 식별 띄어쓰기</p> <hr/> <p>신조어추출 오타자교정 이벤트인식 시맨틱분석</p>	<p>RDF저장소 클래스검색 속성검색 인스턴스 검색 SP 조회 PO 조회 SO 조회 SPARQL Endpoint 질의 상의어, 하의어, 동의어, 유의어 의미망 경로 조회</p> <hr/> <p>데이터변환(Data2RDF) 지식변환 NLQ to SPARQL 지식임베딩</p>	<p>심층 질의응답 사용자의도이해(NLU) MRC 기반 질의응답 음성 질의 응답 음성 인식 음성 합성 이미지 분류</p> <hr/> <p>자동번역 이미지인식 이미지 캡셔닝 얼굴인식 객체인식 선호학습추천</p>	<p>Class 검색 Class 정보조회 Property 검색 Property 정보조회 Instance 검색 Instance 정보조회 Instance 시간정보조회 Instance 공간정보조회 Instance Type 추론 Relation Type 추론 시간 추론 1 시간 추론 2 Object 검색 Subject 검색 Property 검색</p>
					

01 | 자연어 처리 (Natural Language Processing, NLP)

다양한 자연어 처리 Applications

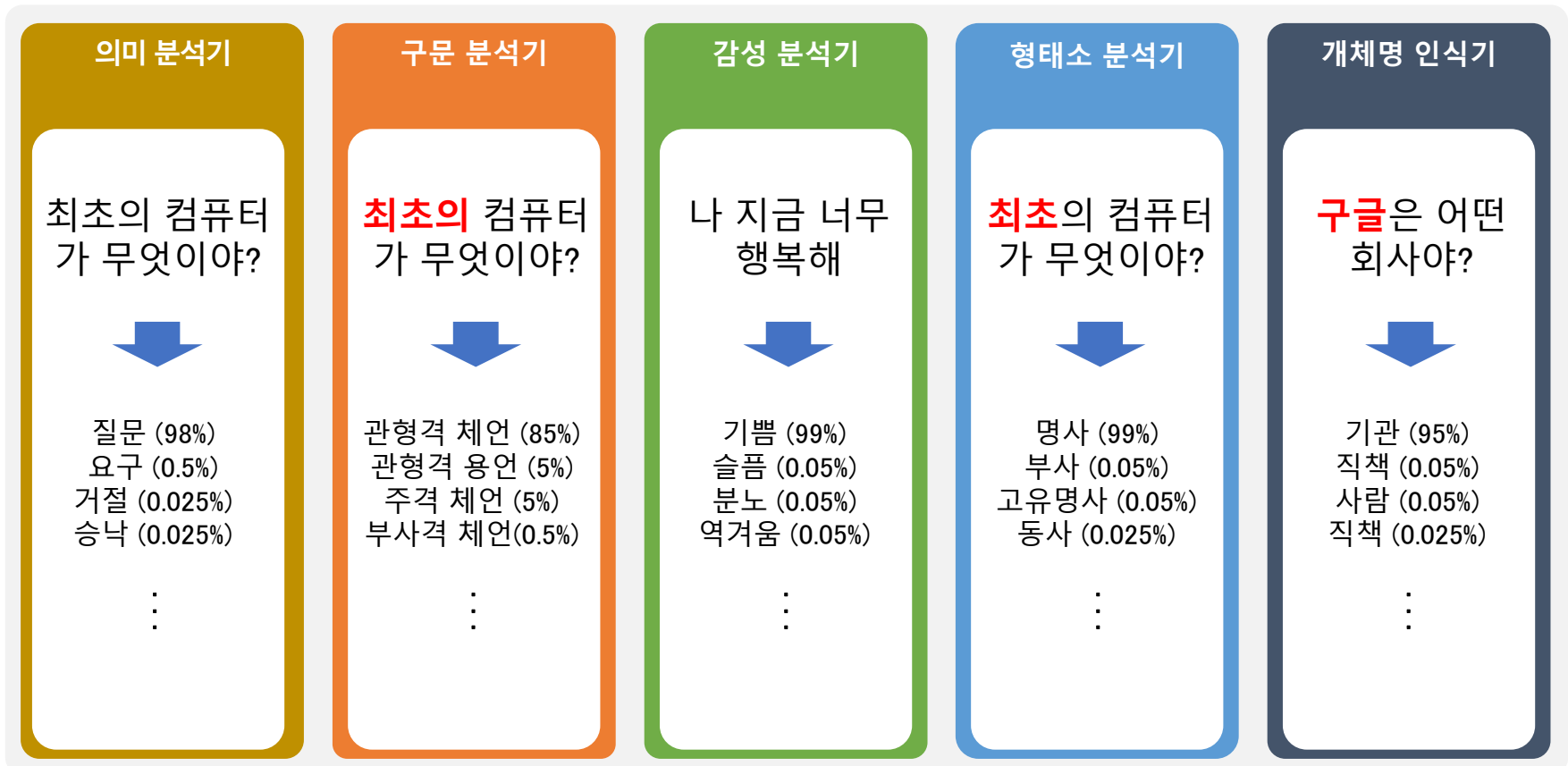
챗봇
+
음성 합성
+
감성 분류
+
개체명 인식
+
추천 시스템
+
기계 독해
+
지식 그래프
+
관계 추출
+
플러그인
⋮

자그마치 9개 이상의 NLP 기술이 합쳐진 컴비네이션



다양한 자연어 처리 Applications

- 형태소 분석, 문서 분류, 개체명 인식 등, 대부분의 자연어 처리 문제는 '분류'의 문제



특징 추출과 분류

- '분류'를 위해선 데이터를 수학적으로 표현
- 먼저, 분류 대상의 특징 (Feature)을 파악 (Feature extraction)

분류 대상

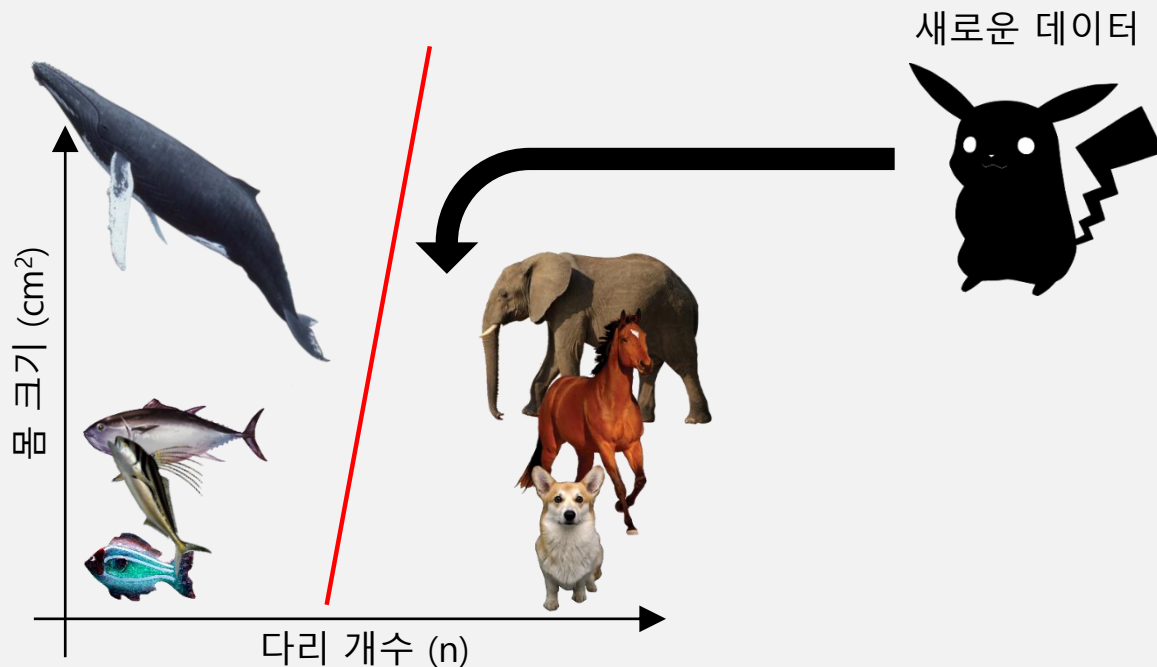


분류 대상의 특징

크기가 다양
다리의 개수가 다양

특징 추출과 분류

- 분류 대상의 특징 (Feature)를 기준으로, 분류 대상을 **그래프 위에 표현** 가능
- 분류 대상들의 경계를 수학적으로 나눌 수 있음 (Classification)
- 새로운 데이터 역시 특징을 기준으로 그래프에 표현하면, 어떤 그룹과 유사한지 파악 가능



자연어에서의 특징 추출과 분류

- 과거에는 사람이 직접 특징 (Feature)를 파악해서 분류
- 실제 복잡한 문제들에선 분류 대상의 특징을 사람이 파악하기 어려울 수 있음
- 이러한 특징을 컴퓨터가 스스로 찾고 (Feature extraction), 스스로 분류 (Classification) 하는 것이 '기계학습'의 핵심 기술

분류 대상	나는 대한민국에서 태어났다	최초의 컴퓨터는 누가 만들었어?
	이순신은 조선 중기의 무신이다	아이유 노래 틀어줘
	나 지금 너무 행복해	내일 날씨 알려줘
		지금 몇 시야?



분류 대상의 특징



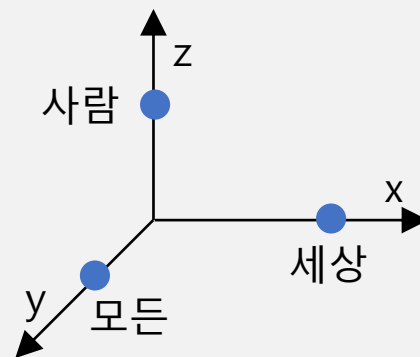
Word embedding – Word2Vec

- 자연어를 어떻게 좌표평면 위에 표현할 수 있을까?
- 가장 단순한 표현 방법은 one-hot encoding 방식 → Sparse representation

세상 모든 사람

단어	Vector
세상	[1, 0, 0]
모든	[0, 1, 0]
사람	[0, 0, 1]

n개의 단어는 n차원의 벡터로 표현



단어 벡터가 sparse해서 단어가 가지는 '**의미**'를 벡터 공간에 표현 불가능

Word embedding – Word2Vec

- Word2vec (word to vector) 알고리즘: 자연어 (특히, 단어) 의 의미를 벡터 공간에 임베딩
- 한 단어의 주변 단어들을 통해, 그 단어의 의미를 파악

جرو

كلب



컴퓨터가 볼 때, 자연어는 우리가 보는 아랍어처럼 '기호' 로만 보일 뿐!

Word embedding – Word2Vec

- Word2vec (word to vector) 알고리즘: 자연어 (특히, 단어) 의 의미를 벡터 공간에 임베딩
- 한 단어의 주변 단어들을 통해, 그 단어의 의미를 파악

جرو
(개) 가 멍멍! 하고 짖었다.

كلب
(강아지) 가 멍멍! 하고 짖었다.



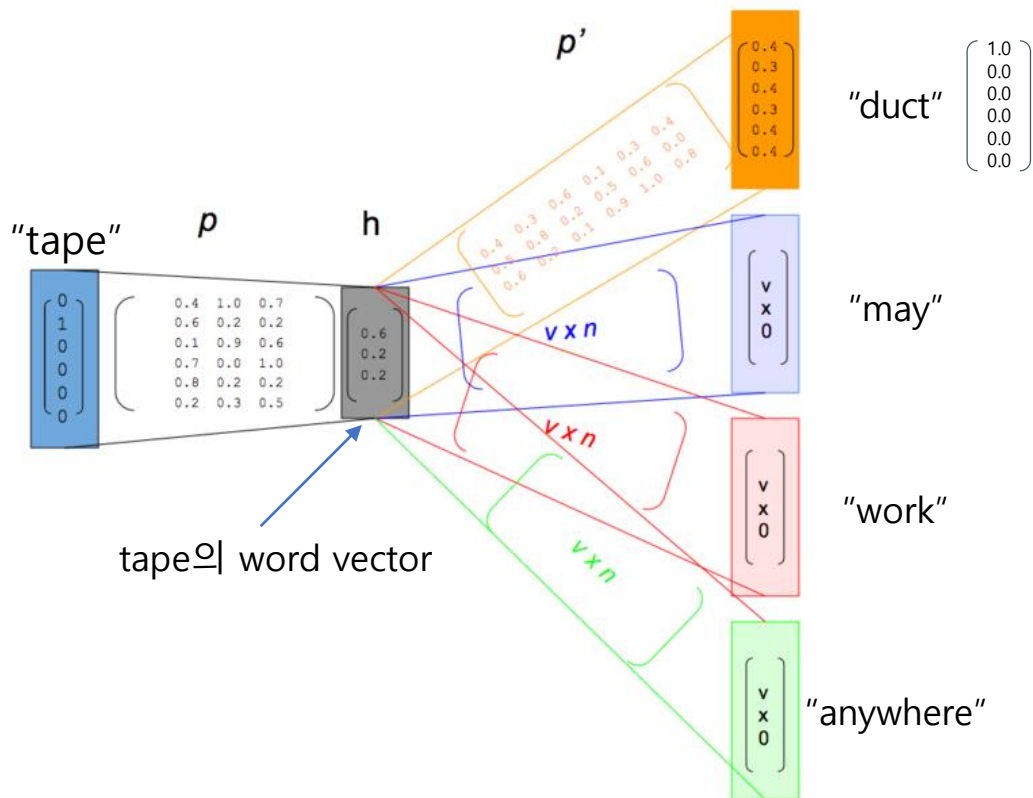
جرو와 كلب가 무슨 뜻인지는 모르겠지만, 주변 단어 형태가 비슷하니 의미도 비슷할 것이다!

Word embedding – Word2Vec

- Word2vec 알고리즘은 주변부의 단어를 예측하는 방식으로 학습 (Skip-gram 방식)
- 단어에 대한 dense vector를 얻을 수 있음

"Duct tape may works anywhere"

Word	One-hot-vector
"duct"	[1, 0, 0, 0, 0]
"tape"	[0, 1, 0, 0, 0]
"may"	[0, 0, 1, 0, 0]
"work"	[0, 0, 0, 1, 0]
"anywhere"	[0, 0, 0, 0, 1]



Word embedding – Word2Vec

- Word embedding의 방법론에 따른 특징

Sparse representation

- One-hot encoding
- n 개의 단어에 대한 n 차원의 벡터
- 단어가 커질 수록 무한대 차원의 벡터가 생성
- 주로 신경망의 입력단에 사용 (신경망이 임베딩 과정을 대체. e.g. `tf.layers.Embedding`)
- 의미 유추 불가능
- 차원의 저주 (curse of dimensionality): 차원이 무한대로 커지면 정보 추출이 어려워짐
- One-hot vector의 차원 축소를 통해 특징을 분류하고자 하는 접근도 있음

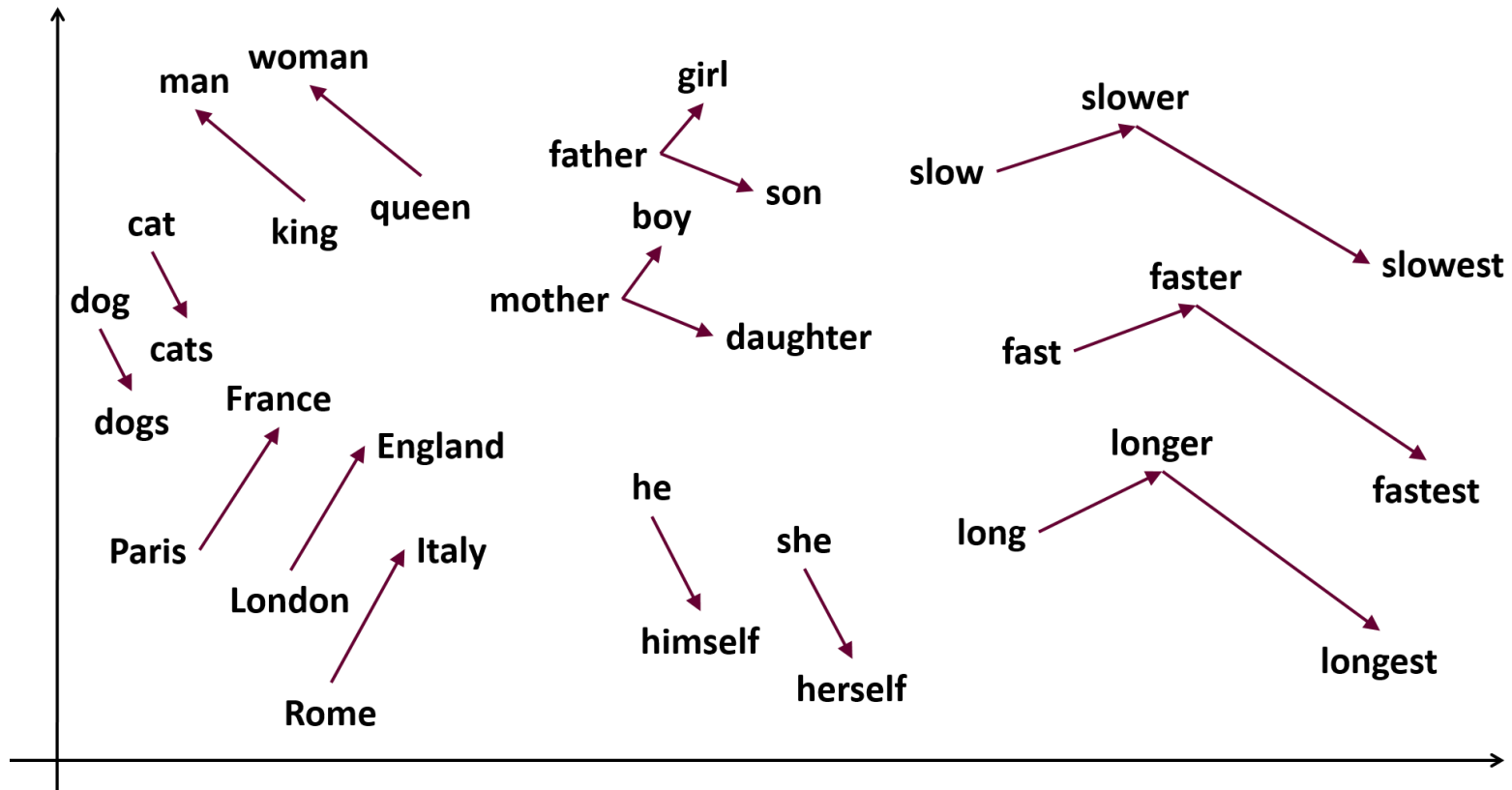
Dense representation

- Word embedding
- 한정된 차원으로 표현 가능
- 의미 관계 유추 가능
- 비지도 학습으로 단어의 의미 학습 가능

Word embedding – Word2Vec

- 단어의 의미가 벡터로 표현됨으로써 벡터 연산이 가능

$$\vec{w}_{king} - \vec{w}_{man} + \vec{w}_{woman} \approx \vec{w}_{queen}$$



Word embedding – Word2Vec

- Word embedding 성능 검증 방법
- (<https://github.com/dongjun-Lee/kor2vec>, <https://github.com/SungjoonPark/KoreanWordVectors>)

WordSim353

- 13-16명의 사람이 annotate한 두 단어의 유사도
- 두 단어 벡터의 cosine similarity를 구한 후 정답과 Spearman's rank-order correlation값을 획득
- 경험상 0.7 이상의 값이 나오면 embedding이 잘 수행 됨

사랑	섹스	6.77
호랑이	고양이	7.35
호랑이	호랑이	10
책	종이	7.46
컴퓨터	키보드	7.62
컴퓨터	인터넷	7.58

⋮

Semantic/Syntactic analogy

- Semantic analogy

대한민국 – 서울 + 도쿄	일본
남동생 – 여동생 + 신부	신랑
왕 – 여왕 + 여동생	남동생
간디 – 인도 + 갈릴레이	이탈리아
베토벤 – 작곡가 + 단테	시인

- Syntactic analogy

밥 – 밥을 + 물을	물
여행이 – 여행은 + 흡연은	흡연이
보도했다 – 보도했습니다 + 알았습니다	알았다
마시다 – 마셨다 + 뽐습니다	뽐았습니다
오다 – 왔다 + 공부했다	공부하다

Word embedding – Word2Vec

• 재미로 보는 Word2Vec analogy test (<https://blog.naver.com/saltluxmarketing/221607368769>)

[할아버지 - 할머니 + 농구] = [배구]
 [할아버지 - 할머니 + 노트북] = [태블릿]
 [할아버지 - 할머니 + 선풍기] = [청소기]
 [할아버지 - 할머니 + 바다] = [숲]
 [할아버지 - 할머니 + 연필] = [볼펜]
 [할아버지 - 할머니 + 파도] = [안개]
 [할아버지 - 할머니 + 물] = [기름]
 [할아버지 - 할머니 + 버스] = [택시]
 [할아버지 - 할머니 + 겨울] = [여름]
 [할아버지 - 할머니 + 신] = [환상]
 [할아버지 - 할머니 + 커피] = [와인]
 [할아버지 - 할머니 + 밥] = [키스]
 [할아버지 - 할머니 + 사탕] = [과자]
 [할아버지 - 할머니 + 소고기] = [닭고기]
 [할아버지 - 할머니 + 치킨] = [피자]
 [할아버지 - 할머니 + 손] = [주먹]
 [할아버지 - 할머니 + 초록색] = [노란색]
 [할아버지 - 할머니 + 기업] = [투자]
 [할아버지 - 할머니 + 사랑] = [행복]
 [할아버지 - 할머니 + 컴퓨터] = [개발자]
 [할아버지 - 할머니 + 시간] = [매일]
 [할아버지 - 할머니 + 공부] = [수업]

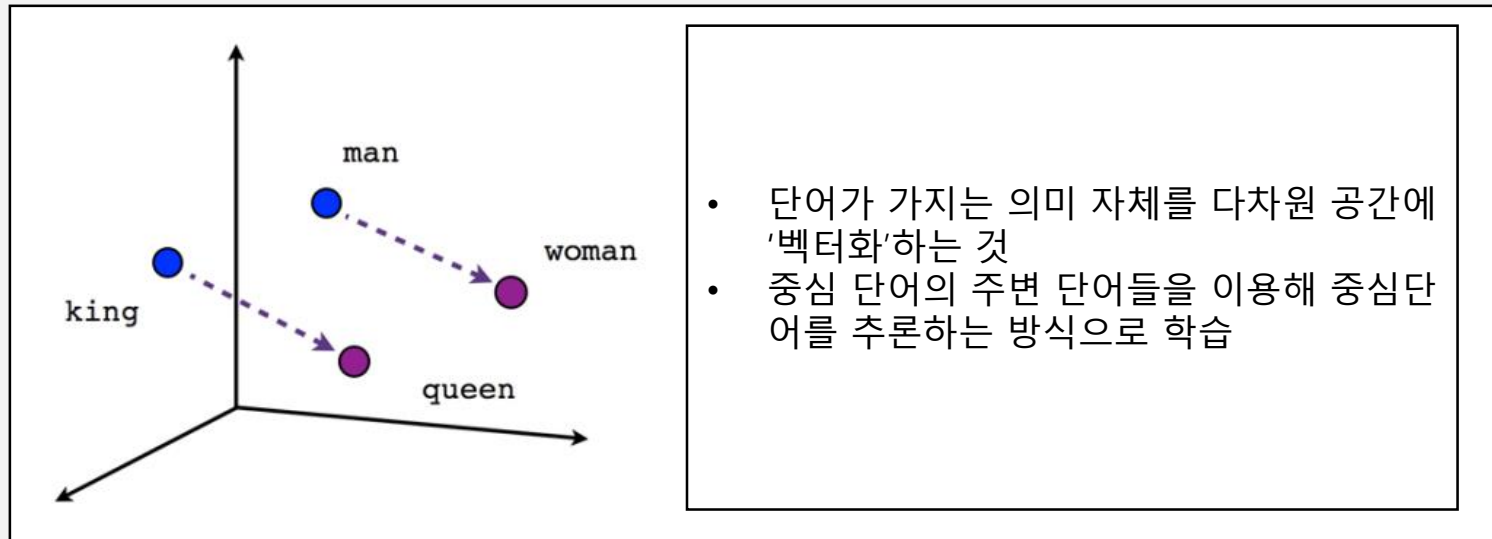
[할아버지 - 할머니 + 인생] = [추억]
 [할아버지 - 할머니 + 기쁨] = [장난]
 [할아버지 - 할머니 + 분노] = [절망]
 [할아버지 - 할머니 + 점심] = [배달]
 [할아버지 - 할머니 + 회의] = [선점]
 [할아버지 - 할머니 + 그림] = [jpg]
 [건물 - 콘크리트 + 사람] = [냄새]
 [건물 - 콘크리트 + 컴퓨터] = [인터페이스]
 [건물 - 콘크리트 + 사랑] = [언제나]
 [건물 - 콘크리트 + 바다] = [모래]
 [건물 - 콘크리트 + 물] = [녹]
 [건물 - 콘크리트 + 시간] = [속도]
 [건물 - 콘크리트 + 공부] = [적응]
 [건물 - 콘크리트 + 인생] = [미소]
 [건물 - 콘크리트 + 손] = [감]
 [건물 - 콘크리트 + 지식] = [구현]
 [건물 - 콘크리트 + 우정] = [목소리]
 [건물 - 콘크리트 + 세계] = [사이클]
 [건물 - 콘크리트 + 분노] = [상처]
 [건물 - 콘크리트 + 힙합] = [리듬]
 [건물 - 콘크리트 + 배고픔] = [방전]

[건물 - 콘크리트 + 태양] = [빛]
 [건물 - 콘크리트 + 쾌락] = [공명]
 [건물 - 콘크리트 + 데이터] = [프로토콜]
 [건물 - 콘크리트 + 구름] = [얼음]
 [건물 - 콘크리트 + 초록색] = [황색]
 [건물 - 콘크리트 + 자유] = [도파민]
 [여름 - 더위 + 겨울] = [마름]
 [여름 - 더위 + 인간] = [욕구]
 [여름 - 더위 + 바다] = [플랑크톤]
 [여름 - 더위 + 재미] = [자질]
 [선풍기 - 바람 + 눈] = [눈물]
 [사람 - 지능 + 컴퓨터] = [소프트웨어]
 [인생 - 사람 + 컴퓨터] = [관리자]
 [그림 - 연필 + 영화] = [스타]
 [오케스트라 - 바이올린 + 인간] = [육체]
 [손 - 박수 + 발] = [달리기]
 [삼겹살 - 소주 + 맥주] = [햄]

Word embedding – Word2Vec 실습

```
1  from gensim.models.word2vec import Word2Vec
2  import gensim
3
4  path = 'train_corpus.txt'
5  sentences = gensim.models.word2vec.Text8Corpus(path)
6
7  model = Word2Vec(sentences, min_count=5, size=100, window=5)
8  model.save('w2v_model')
9
10 saved_model = Word2Vec.load('w2v_model')
11
12 word_vector = saved_model['강아지']
13
14 saved_model.similarity('강아지', '멍멍이')
15
16 saved_model.similar_by_word('강아지')
```


Word embedding – Word2Vec



장점

- 단어간의 유사도 측정에 용이
- 단어간의 관계 파악에 용이
- 벡터 연산을 통한 추론이 가능 (e.g. 한국 - 서울 + 도쿄 = ?)

단점

- 단어의 **subword information** 무시 (e.g. 서울 vs 서울시 vs 고양시)
- Out of vocabulary (OOV)에서 적용 불가능

Word embedding – FastText

- 한국어는 다양한 용언 형태를 가짐
- Word2Vec의 경우, 다양한 용언 표현들이 서로 독립된 vocab으로 관리

동사 원형: **모르다**

모르네	모르기까지	모르겠으나	몰라야	몰랐다면	몰랐겠으나
모르데	모르기를	모르겠으면	몰라요	몰랐다면	몰랐겠으면
모르지	모르기는	모르겠으면서	몰라라	몰랐을	몰랐겠으면서
모르더라	모르기도	모르겠거나	몰랐다	몰랐을까	몰랐겠거나
모르리라	모르기만	모르겠거든	몰랐네	몰랐을지	몰랐겠거든
모르는구나	모르는	모르겠는데	몰랐지	몰랐을지도	몰랐겠는데
모르잖아	모르던	모르겠지만	몰랐더라	몰랐어	몰랐겠지만
모르려나	모른	모르겠더라도	몰랐으리라	몰랐어도	몰랐겠더라도
모르니	모른다	모르겠다가도	몰랐구나	몰랐어야	몰랐겠다가도
모르고	모른다면	모르겠던	몰랐잖아	몰랐어요	몰랐겠던
모르나	모른다면	모르겠다면	몰랐으려나	몰랐더라면	몰랐겠다면
모르면	모른답시고	모르겠다면	몰랐으니	몰랐더라도	몰랐겠다면
모르면서	모르겠다	모를까	몰랐거나	몰랐겠다	몰랐겠어
모르거든	모르겠네	모를지	몰랐거든	몰랐겠네	몰랐겠어도
모르는데	모르겠지	모를지도	몰랐는데	몰랐겠지	몰랐겠어서
모르지만	모르겠더라	모를수록	몰랐지만	몰랐겠더라	몰랐겠어야
모르더라도	모르겠구나	몰라	몰랐더라도	몰랐겠구나	몰랐겠어요
모르다가도	모르겠니	몰라도	몰랐다가도	몰랐겠니	몰랐겠더라면
모르기조차	모르겠고	몰라서	몰랐던	몰랐겠고	몰랐겠더라도

Word embedding – FastText

Fasttext

- Facebook research에서 공개한 open source library (<https://research.fb.com/fasttext/>, fasttext.cc)
- C++11

Training

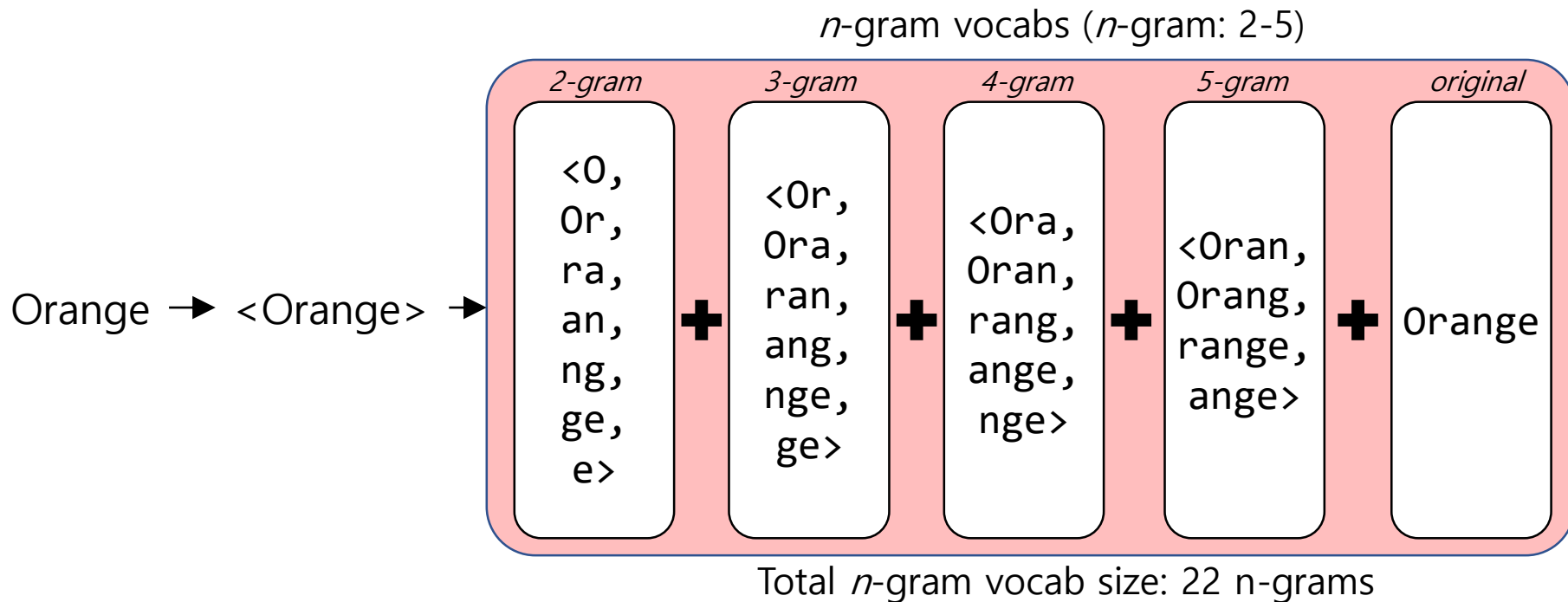
- 기존의 word2vec과 유사하나, 단어를 **n-gram**으로 나누어 학습을 수행
- **n-gram**의 범위가 **2-5**일 때, 단어를 다음과 같이 분리하여 학습함
"assumption" = {as, ss, su,, ass, ssu, sum, ump, mpt,, ption, **assumption**}
- 이 때, *n*-gram으로 나뉜 단어는 사전에 들어가지 않으며, 별도의 *n*-gram vector를 형성함

Testing

- 입력 단어가 vocabulary에 있을 경우, word2vec과 마찬가지로 해당 단어의 word vector를 return함
- 만약 OOV일 경우, 입력 단어의 *n*-gram vector들의 합산을 return함

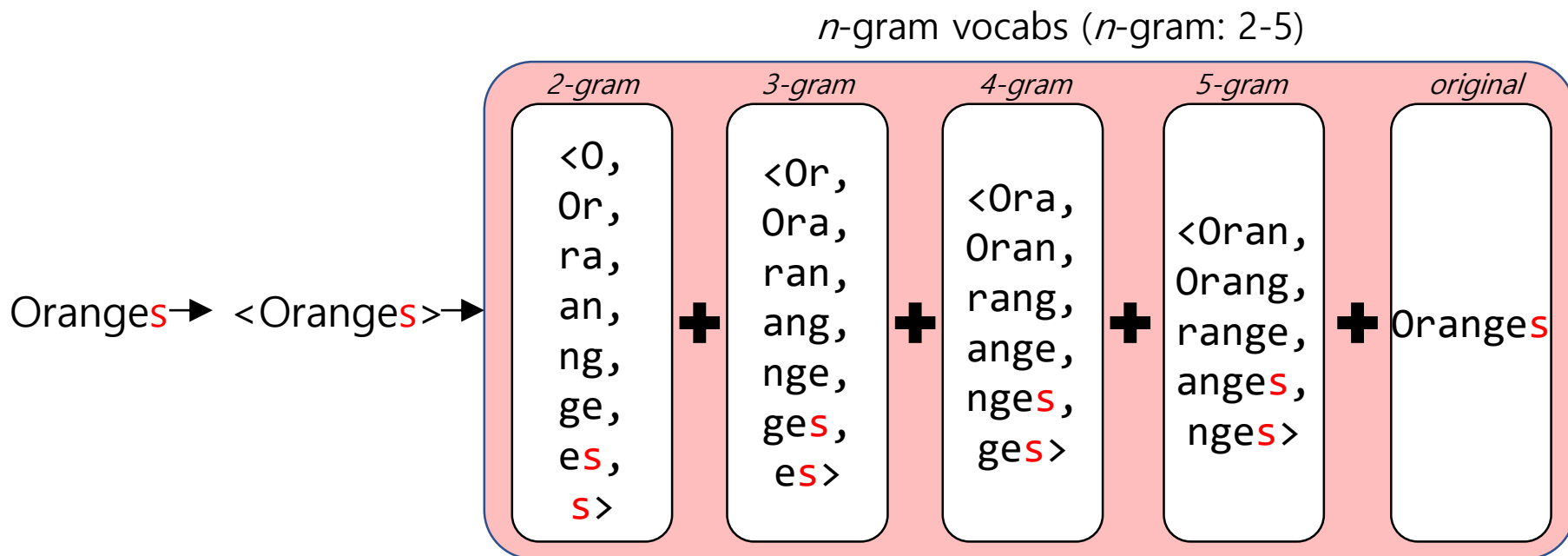
Word embedding – FastText

- FastText는 단어를 n -gram으로 분리한 후, 모든 n -gram vector를 합산한 후 평균을 통해 단어 벡터를 획득



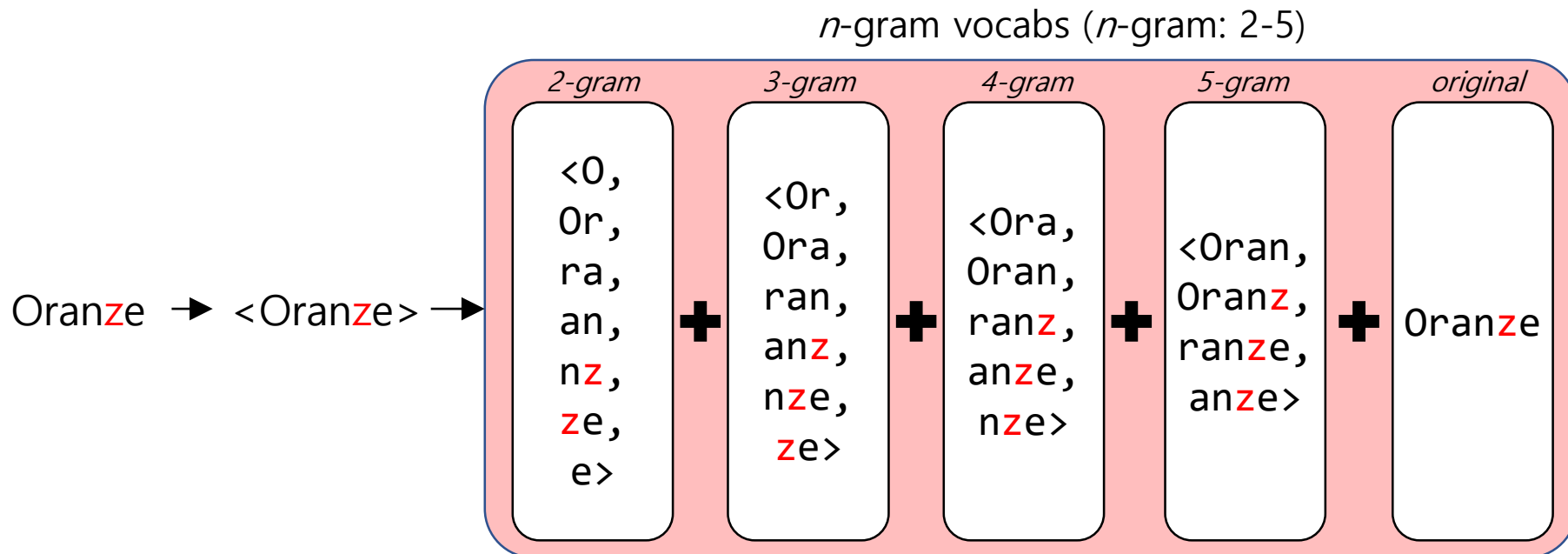
Word embedding – FastText

- FastText는 단어를 n -gram으로 분리한 후, 모든 n -gram vector를 합산한 후 평균을 통해 단어 벡터를 획득



Word embedding – FastText

- 오타자 입력에 대해서도 본래 단어와 유사한 n -gram이 많아, 유사한 단어 벡터를 획득 가능



FastText와 Word2Vec의 비교

- 오타자, OOV, 등장 회수가 적은 학습 단어에 대해서 강세

(https://link.springer.com/chapter/10.1007/978-3-030-12385-7_3)

Input word	DF	Model	Most similar (1)	Most similar (2)	Most similar (3)	Most similar (4)	Most similar (5)
파일	10146	Word2vec	프로토콜	애플리케이션	url	디렉터리	포맷
		Fasttext	파일	확장자	음악파일	포맷	디렉터리
원인	11589	Word2vec	부작용	현상	요인	증상	질병
		Fasttext	주원인	초래	요인	직접	주요인
태생지	8	Word2vec	부타	구티에레즈	보아뱀	올림픽코	집사장
		Fasttext	생지	탄생지	출생지	발생지	무생지
미스코리아	11	Word2vec	치평요람	神檀實記	컬투	방학기	김현승
		Fasttext	믹스코리아	라이코스코리아	악스코리아	보이스코리아	포브스코리아

- Document frequency (DF) 가 낮을 경우, **word2vec**은 유의미한 단어를 찾을 수 없음
- **Fasttext**의 경우, **subword information**이 유사한 단어를 찾았음

Advanced FastText 모델

- 오타자, OOV, 등장 회수가 적은 학습 단어에 대해서 강세

(https://link.springer.com/chapter/10.1007/978-3-030-12385-7_3)

Input word	FastText Model	Most similar words in range of top 3		
페널티 (Penalty) OOV word	Baseline	리날디 (Rinaldi)	페레티 (Ferretti)	마세티 (Machete)
	Jamo-advanced	페널티골 (Penalty goal)	페널티 (Penalty)	드로바 (Drogba)
나프탈렌 (Naphthalene) OOV word	Baseline	야렌 (Yaren)	콜루바라 (Kolubara)	몽클로아아라바카 (Moncloa-Aravaca)
	Jamo-advanced	나프탈렌 (Naphthalene)	테레프탈산 (Terephthalic acid)	아디프산 (Adipic acid)
스테이크 (Steak) OOV word	Baseline	스테너프 (Stanhope)	스테너드 (Stannard)	화이트스네이크 (White Snake)
	Jamo-advanced	롱테이크 (Long take)	비프스테이크 (Beefsteak)	스테이크 (Steak)

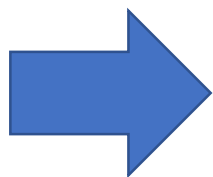
Word embedding – FastText 실습

```
1 from gensim.models.word2vec import FastText
2 import gensim
3
4 path = 'train_corpus.txt'
5 sentences = gensim.models.word2vec.Text8Corpus(path)
6
7 model = FastText(sentences, min_count=5, size=100, window=5)
8 model.save('ft_model')
9
10 saved_model = FastText.load('ft_model')
11
12 word_vector = saved_model['강아지']
13
14 saved_model.similarity('강아지', '멍멍이')
15
16 saved_model.similar_by_word('강아지')
```

01 | 자연어 처리 (Natural Language Processing, NLP)

Word embedding의 활용

- 다른 자연어처리 모델의 입력으로 사용
(e.g. 학습 데이터의 양이 매우 적은 감성분석을 수행할 경우, 학습 데이터만으로는 특성 추출 불가능)
- 토픽 키워드 (<https://www.adams.ai/apiPage?deeptopicrank>)



BERT 관련 토픽 키워드 추출



Word embedding 방식의 한계점

- Word2Vec이나 FastText와 같은 word embedding 방식은 동형어, 다의어 등에 대해선 embedding 성능이 좋지 못하다는 단점이 있음
- 주변 단어를 통해 학습이 이루어지기 때문에, '문맥'을 고려할 수 없음

Account

- | | |
|--------------------------------------------------------------------------------|----------|
| 1) I account him to be my friend | ~라고 생각하다 |
| 2) He is angry on account of being excluded from the invitation | 이유, 근거 |
| 3) I have an account with the First National Bank | ~때문에 |
| 4) The police wrote an account of the accident | 보고서 |
| 5) Your check has been properly credited, and your account is now full? | 계좌 |

언어 모델

- 언어 모델의 개념과, BERT가 탄생하기 이전의 언어모델들을 살펴보겠습니다

모델이란?

[IT용어]모형(model)

1. ① 어떤 상황이나 물체 등 연구 대상 주제를 도면이나 사진 등 화상을 사용하거나 수식이나 악보와 같은 기호를 사용하여 표현한 것. 표현 양식에 따라서 화상 모형(graphical model)이나 기호 모형(symbolic model)이라고 하고, 특히 수학적 기호와 수식을 사용하여 표현한 것을 수학적 모형이라고 한다. 분자(molecule)의 도식 모형, 우주의 물질 분포의 수학적 모형, 기업 경영의 표 계산(숫자적) 모형 등이 있다. 모형을 변경하거나 조작하여 그것이 변형, 수정 또는 조건의 변화에 의해 어떻게 달라지는가를 알아낼 수 있다.

• 모델의 종류

- 일기예보 모델, 데이터 모델, 비즈니스 모델, 물리 모델, 분자 모델 등

• 모델의 특징

- 자연 법칙을 컴퓨터로 모사함으로써 시뮬레이션이 가능
- 이전 state를 기반으로 미래의 state를 예측할 수 있음
(e.g. 습도와 바람 세기 등으로 내일 날씨 예측)
- 즉, 미래의 state를 올바르게 예측하는 방식으로 모델 학습이 가능함

언어 모델

- '자연어'의 법칙을 컴퓨터로 모사한 모델 → 언어 '모델'
- 주어진 단어들로부터 그 다음에 등장한 단어의 확률을 예측하는 방식으로 학습 (이전 state로 미래 state를 예측)
- 다음의 등장할 단어를 잘 예측하는 모델은 그 언어의 특성이 잘 반영된 모델이자, 문맥을 잘 계산하는 좋은 언어 모델



딥러닝을 이용한



딥러닝을 이용한 주가 예측 모델
딥러닝을 이용한 한국어 자동 띄어쓰기
딥러닝을 이용한 자연어 처리
딥러닝을 이용한
딥러닝을 이용한 자연어처리 입문
딥러닝을 이용한 이미지 인식
딥러닝을 이용한 얼굴인식
딥러닝을 이용한 악성코드탐지 연구동향
딥러닝을 이용한 차량 모델의 진동해석 예측
딥러닝을 이용한 영상처리

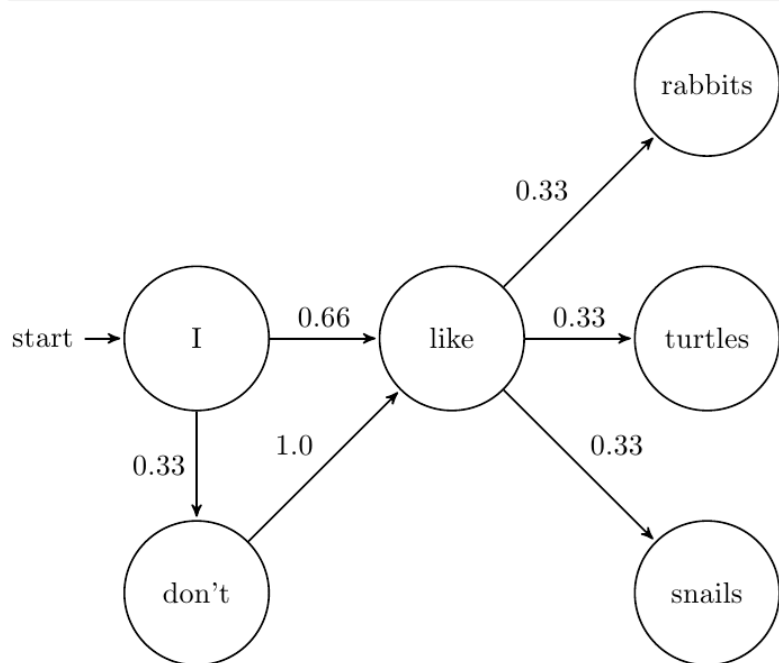
Google 검색

I'm Feeling Lucky

부적절한 예상 검색어 신고

Markov 확률 기반의 언어 모델

- 마코프 체인 모델 (Markov Chain Model)
- 초기의 언어 모델은 다음의 단어나 문장이 나올 확률을 통계와 단어의 n -gram을 기반으로 계산
- 딥러닝 기반의 언어모델은 해당 확률을 최대로 하도록 네트워크를 학습



	I	don't	like	rabbits	turtles	snails
I	0	0.33	0.66	0	0	0
don't	0	0	1.0	0	0	0
like	0	0	0	0.33	0.33	0.33
rabbits	0	0	0	0	0	0
turtles	0	0	0	0	0	0
snails	0	0	0	0	0	0

"I like rabbits"



$$0.66 * 0.33 = 0.22$$

"I like turtles"



$$0.66 * 0.33 = 0.22$$

"I don't like snails"

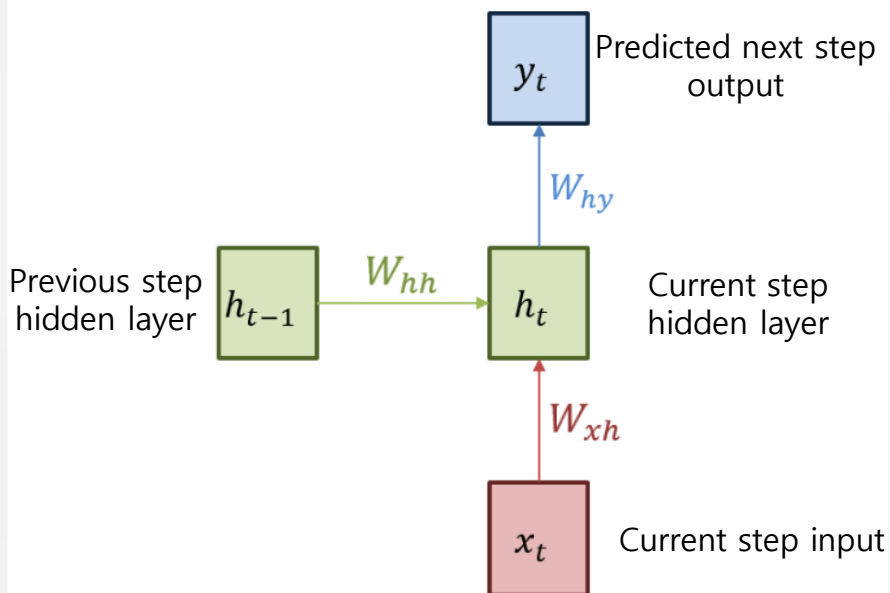


$$0.33 * 1.0 * 0.33 = 0.11$$

Recurrent Neural Network (RNN) 기반의 언어 모델

- RNN은 히든 노드가 방향을 가진 엣지로 연결돼 순환구조를 이루는(directed cycle) 인공신경망의 한 종류
- 이전 state 정보가 다음 state를 예측하는데 사용됨으로써, 시계열 데이터 처리에 특화

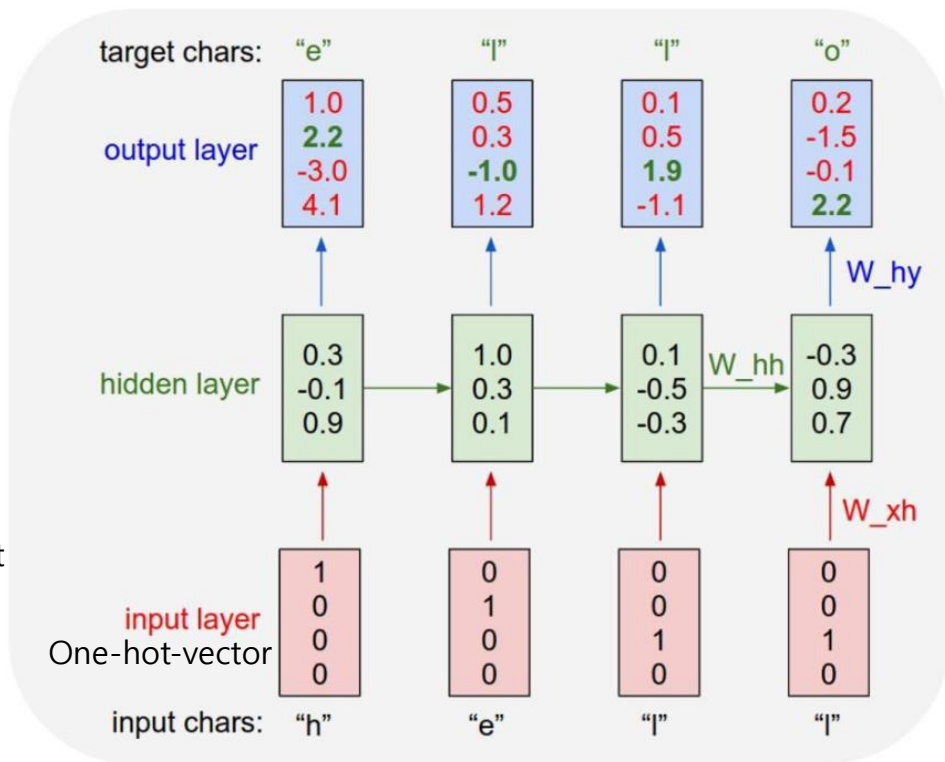
기본적인 RNN의 구조



$$y_t = W_{hy}h_t + b_y$$

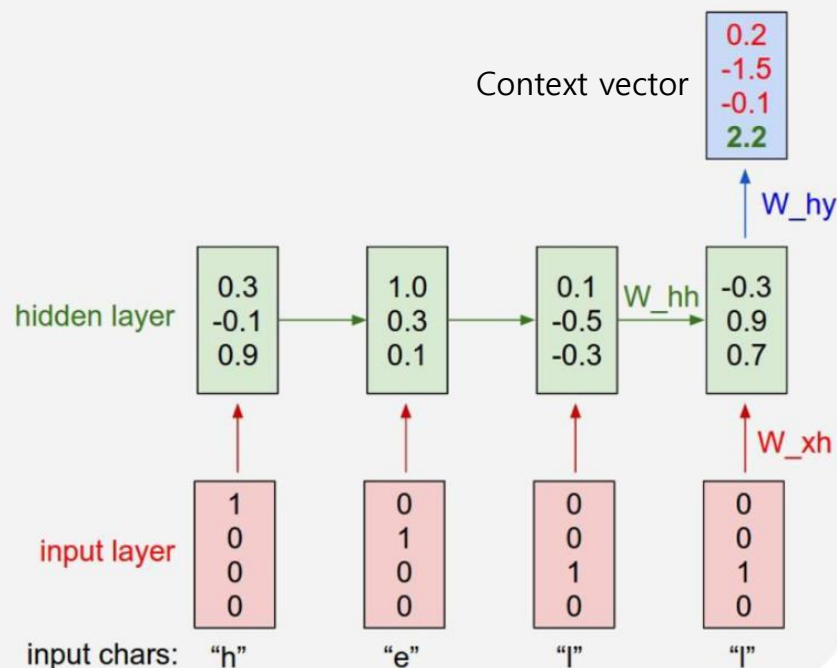
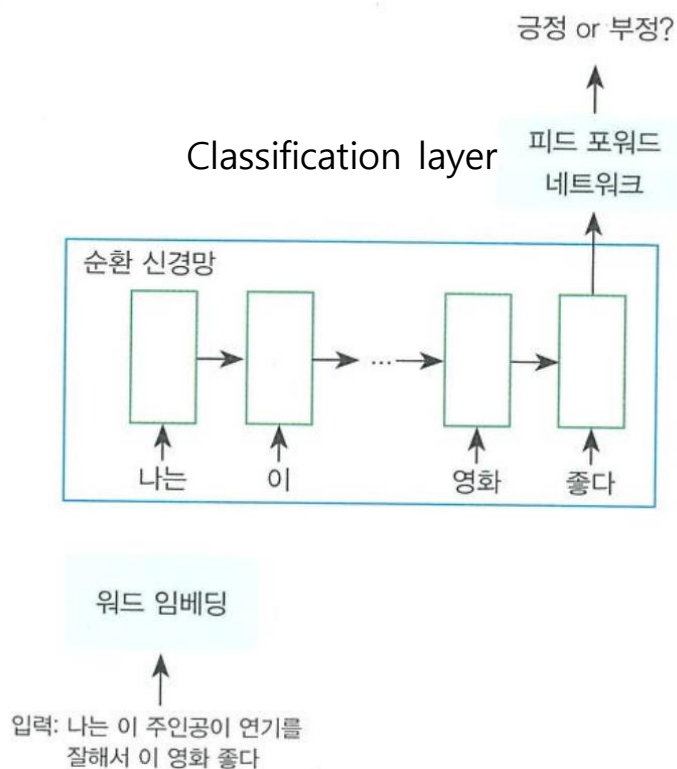
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

다음 문자를 학습하는 RNN encoder



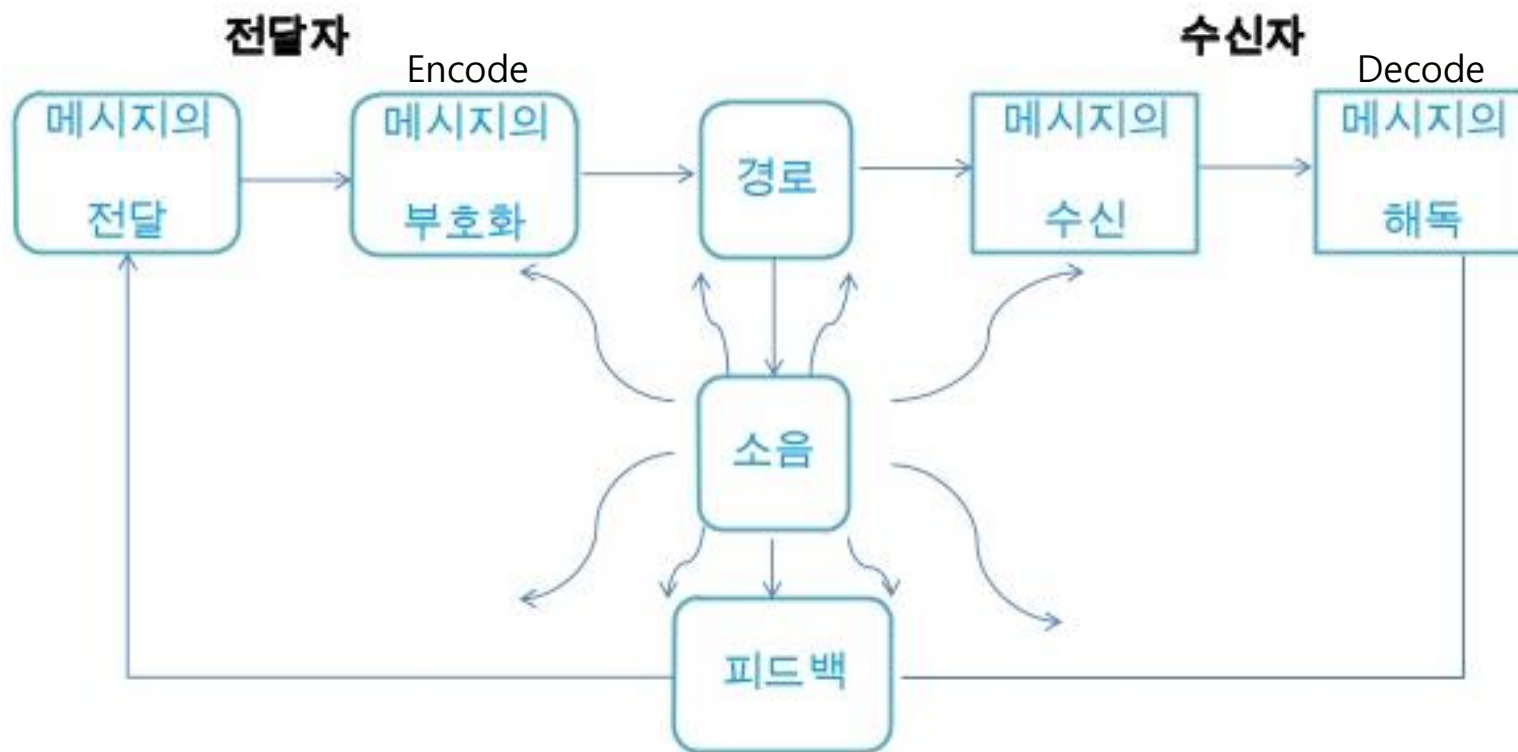
RNN 언어 모델을 이용한 Application

- 마지막 출력은 앞선 단어들의 '문맥' 을 고려해서 만들어진 최종 출력 vector → Context vector
- 출력된 context vector 값에 대해 classification layer를 붙이면 문장 분류를 위한 신경망 모델



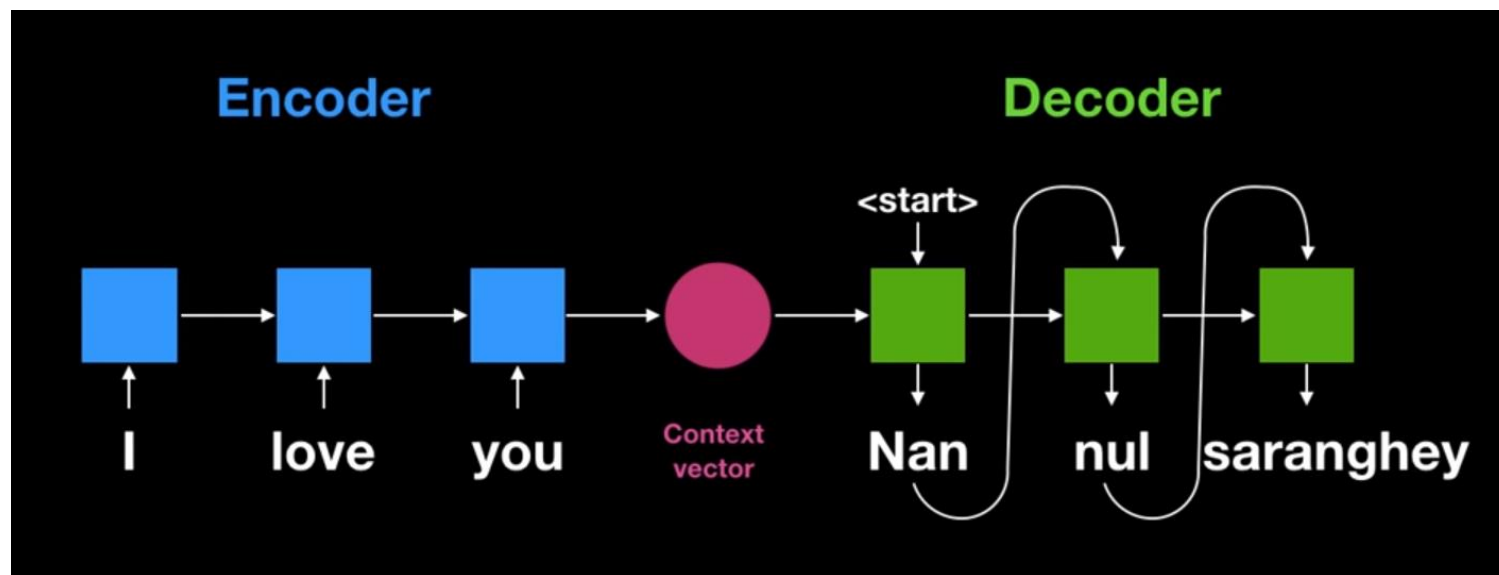
Encoder - Decoder

- RNN을 통해서 잘 encoding 된 언어를 다시 decoding을 하면?



RNN 모델 기반의 Seq2Seq

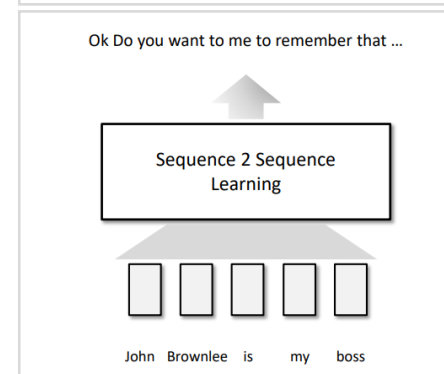
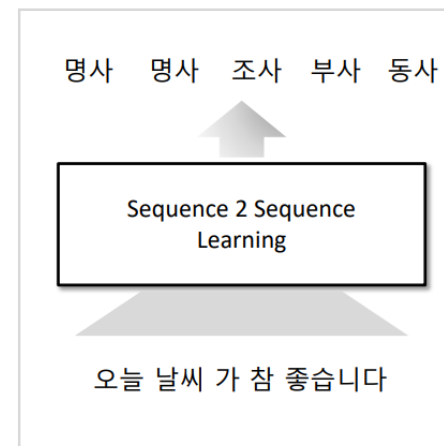
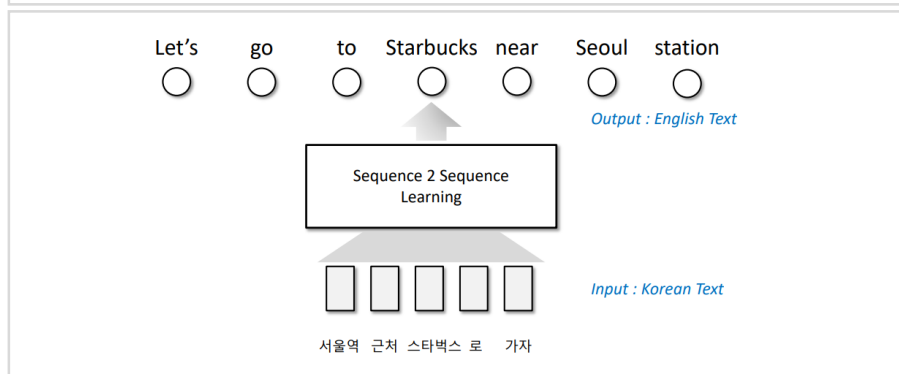
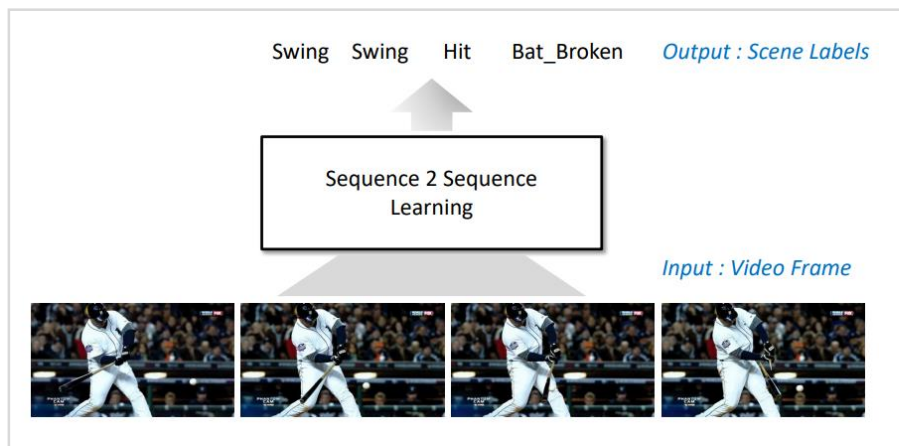
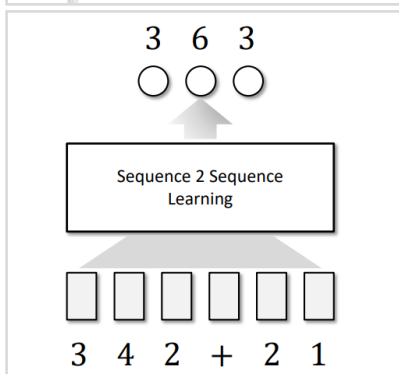
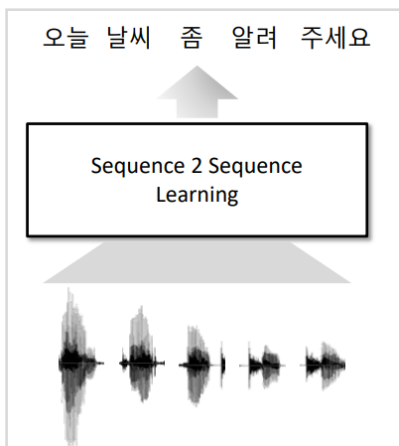
- Seq2Seq (Sequence to Sequence)
- Encoder layer: RNN 구조를 통해 Context vector 를 획득
- Decoder layer: 획득된 Context vector를 입력으로 출력을 예측



* [딥러닝 기계번역] 시퀀스 투 시퀀스 + 어텐션 모델
(<https://www.youtube.com/watch?v=WsQLdu2JMgl>)

Seq2Seq Applications

- Seq2Seq (Sequence to Sequence)
- Encoder layer: Context vector 를 획득
- Decoder layer: 획득된 Context vector를 입력으로 다음 state를 예측

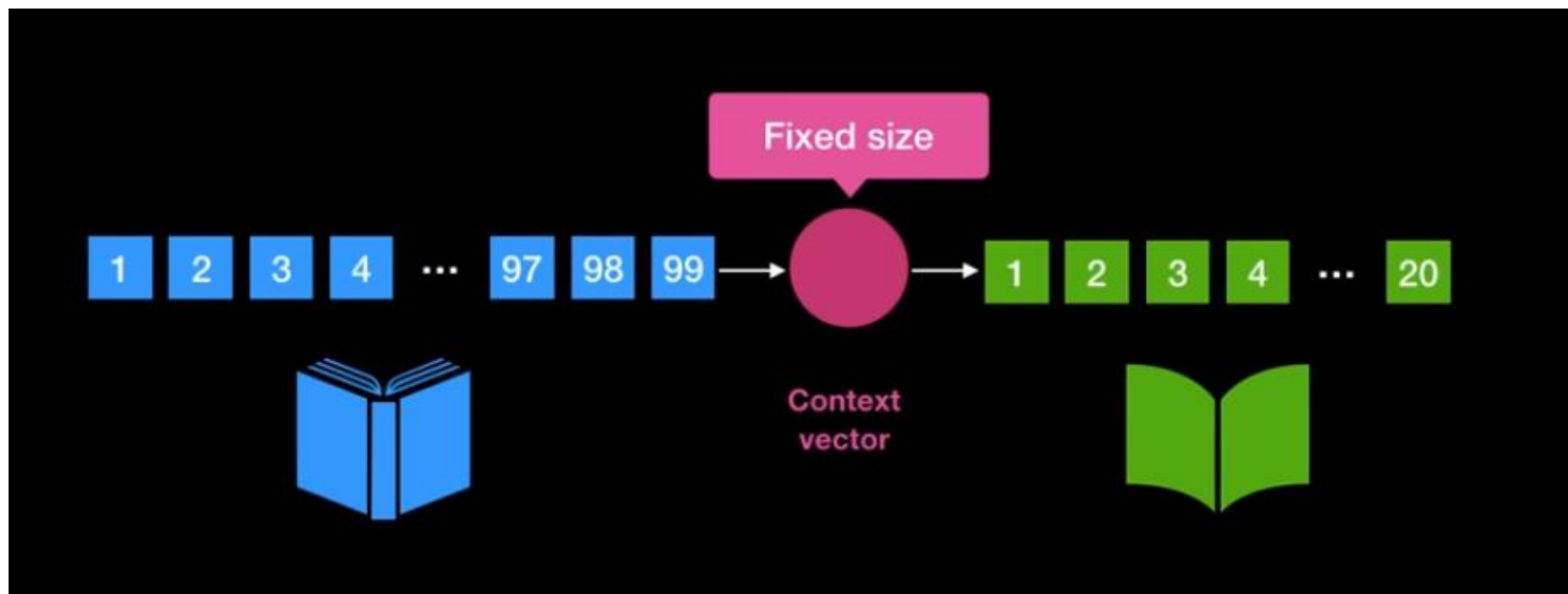


Seq2Seq Applications



RNN 의 구조적 문제점

- 입력 sequence의 길이가 매우 긴 경우, 처음에 나온 token에 대한 정보가 희석
- 고정된 context vector 사이즈로 인해 긴 sequence에 대한 정보를 함축하기 어려움
- 모든 token이 영향을 미치니, 중요하지 않은 token도 영향을 줌



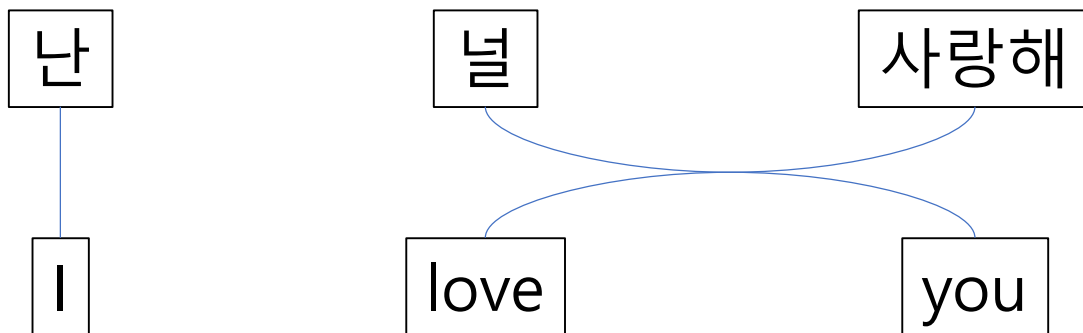
* [딥러닝 기계번역] 시퀀스 투 시퀀스 + 어텐션 모델
(<https://www.youtube.com/watch?v=WsQLdu2JMgl>)



Attention의 탄생!

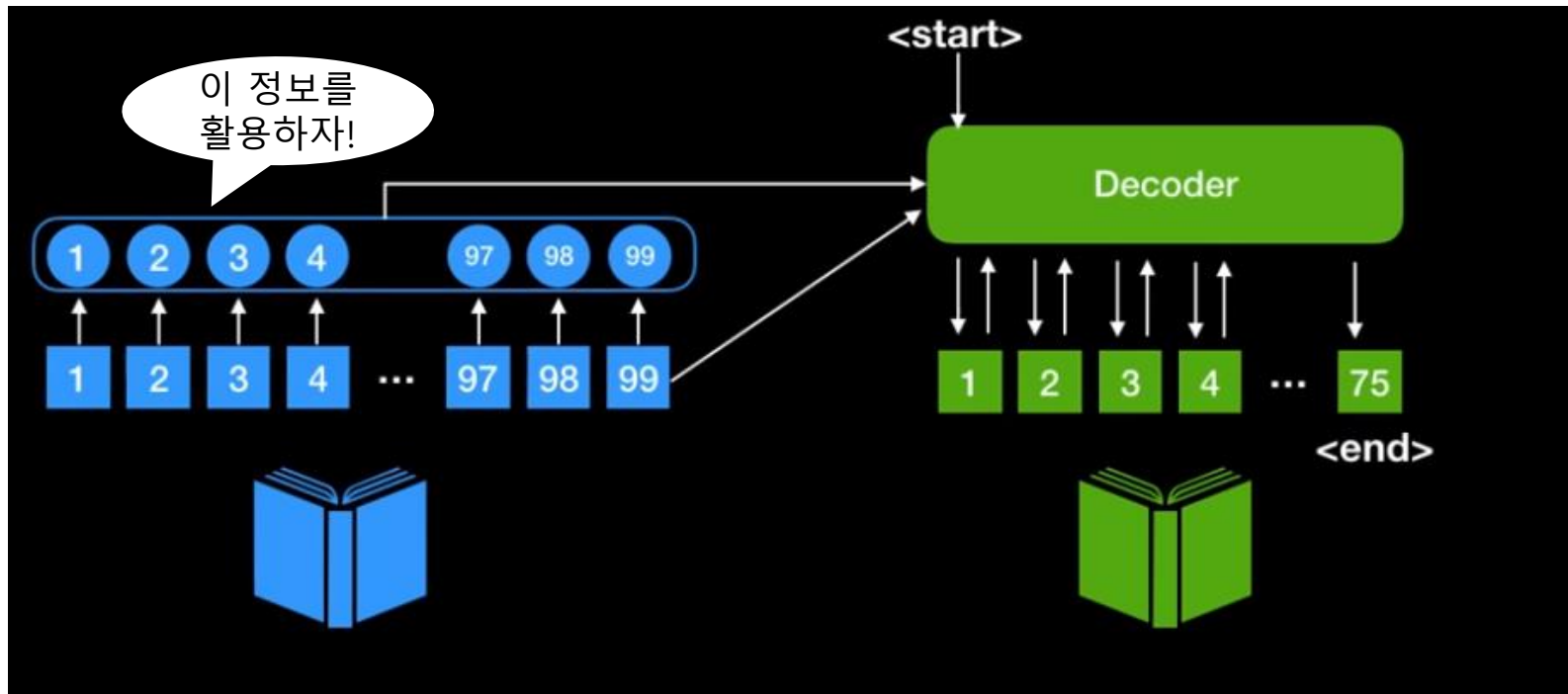
Attention 모델

- 인간이 정보처리를 할 때, 모든 sequence를 고려하면서 정보처리를 하는 것이 아님
- 인간의 정보처리와 마찬가지로, 중요한 feature는 더욱 중요하게 고려하는 것이 Attention의 모티브



Attention 모델

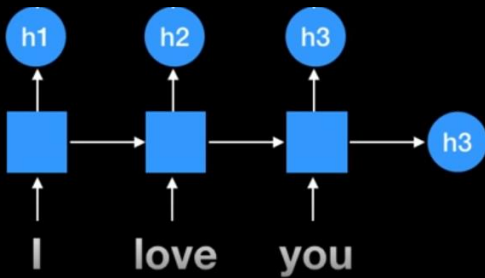
- 기존 Seq2Seq에서는 RNN의 최종 output인 Context vector만을 활용
- Attention에서는 인코더 RNN 셀의 각각 output을 활용
- Decoder에서는 매 step마다 RNN 셀의 output을 이용해 dynamic하게 Context vector를 생성



* [딥러닝 기계번역] 시퀀스 투 시퀀스 + 어텐션 모델
(<https://www.youtube.com/watch?v=WsQLdu2JMgI>)

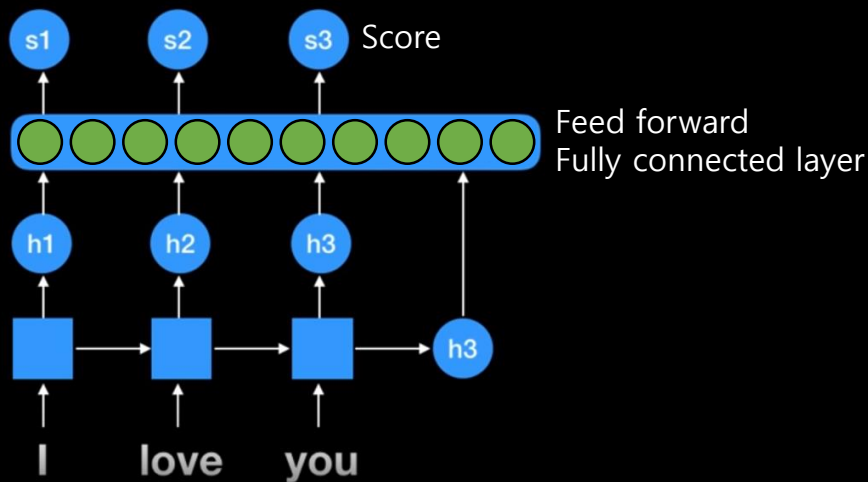
Attention 모델

- 기존 RNN 모델에서 시작



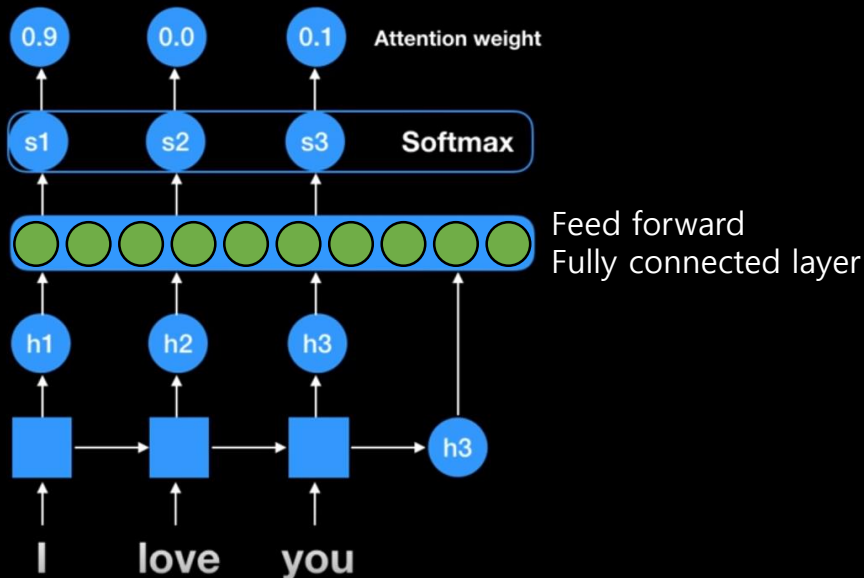
Attention 모델

- 기존 RNN 모델에서 시작
- RNN 셀의 각 output들을 입력으로 하는 Feed forward fully connected layer
- 해당 layer의 output을 각 RNN 셀의 Score로 결정



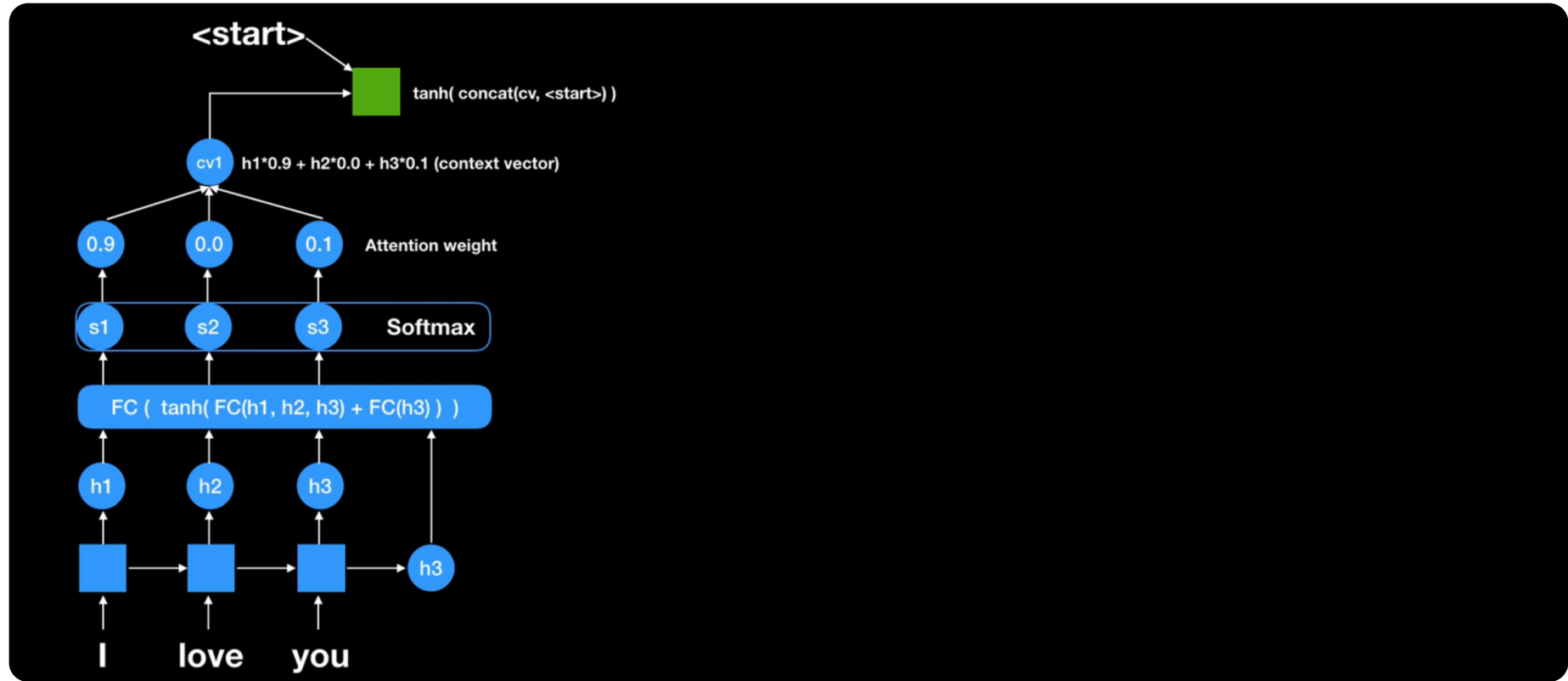
Attention 모델

- 출력된 score에 Softmax를 취함으로써 0-1 사이의 값으로 변환
- 해당 값을 Attention weight로 결정



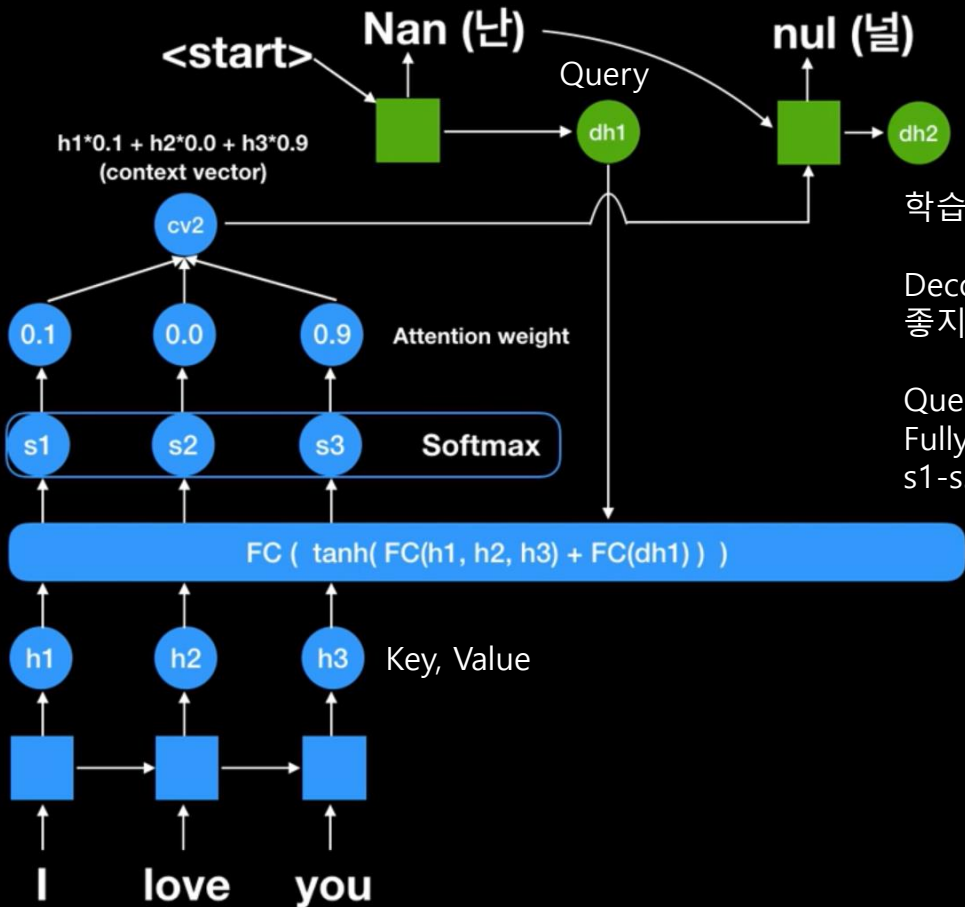
Attention 모델

- 출력된 score에 Softmax를 취함으로써 0-1 사이의 값으로 변환
- 해당 값을 Attention weight로 결정
- Attention weight와 hidden state를 곱해서 Context vector 획득



Attention 모델

- Decoder의 hidden state가 attention weight 계산에 영향을 줌

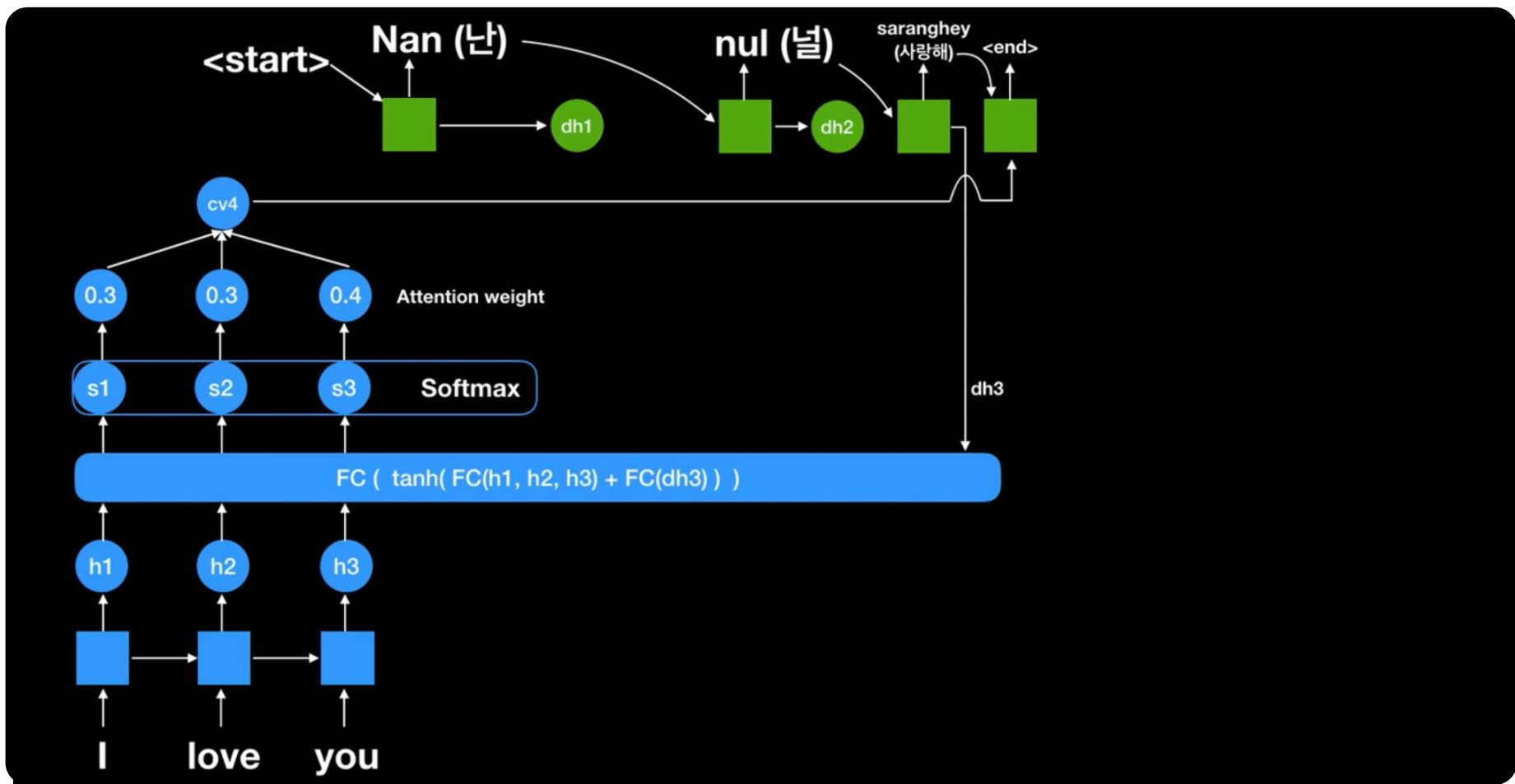


Decoder 결과가 정답과 많이 다르다
 좋지 못한 context vector → 좋지 못한 attention weight

Query인 '난' 과 'I', 'love', 'you'와의 상관관계 조정 필요
 Fully connected feed forward network에서 score를 조정
 s_1-s_3 가 조정됨에 따라 Attention weight도 조정

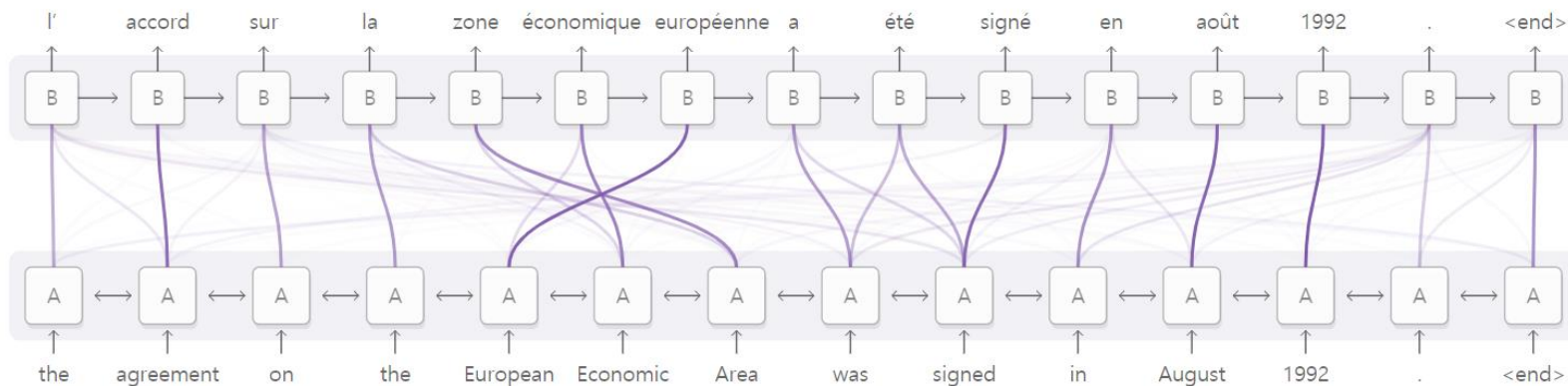
Attention 모델

- Decoder의 hidden state가 attention weight 계산에 영향을 줌

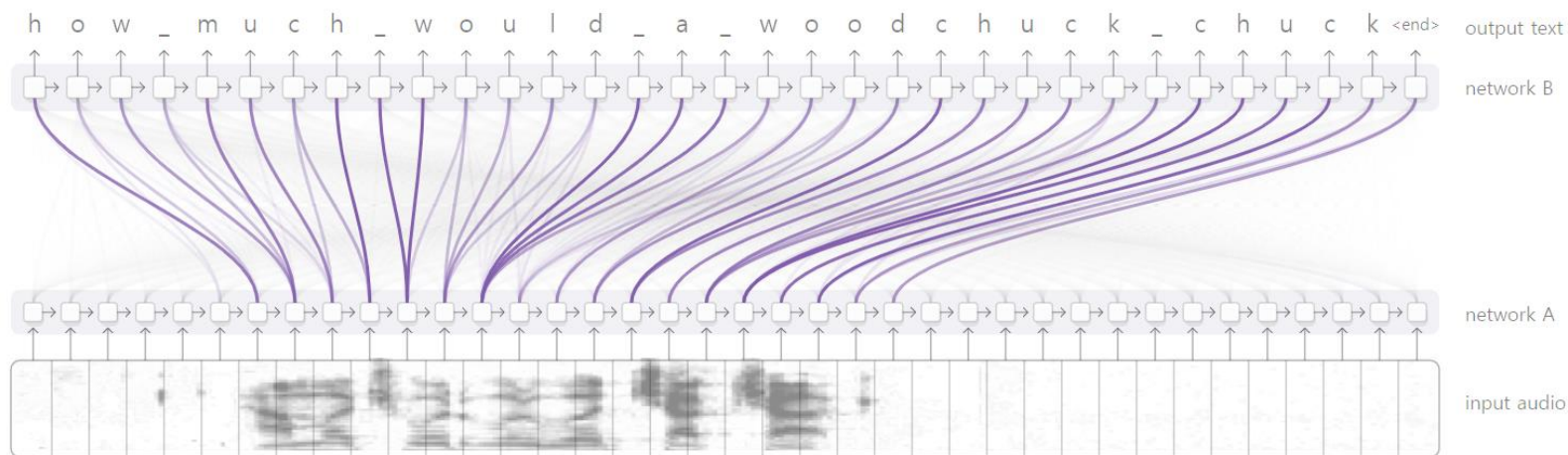


Attention 모델을 이용한 딥러닝의 시각화

- Attention for neural machine translation (NMT)

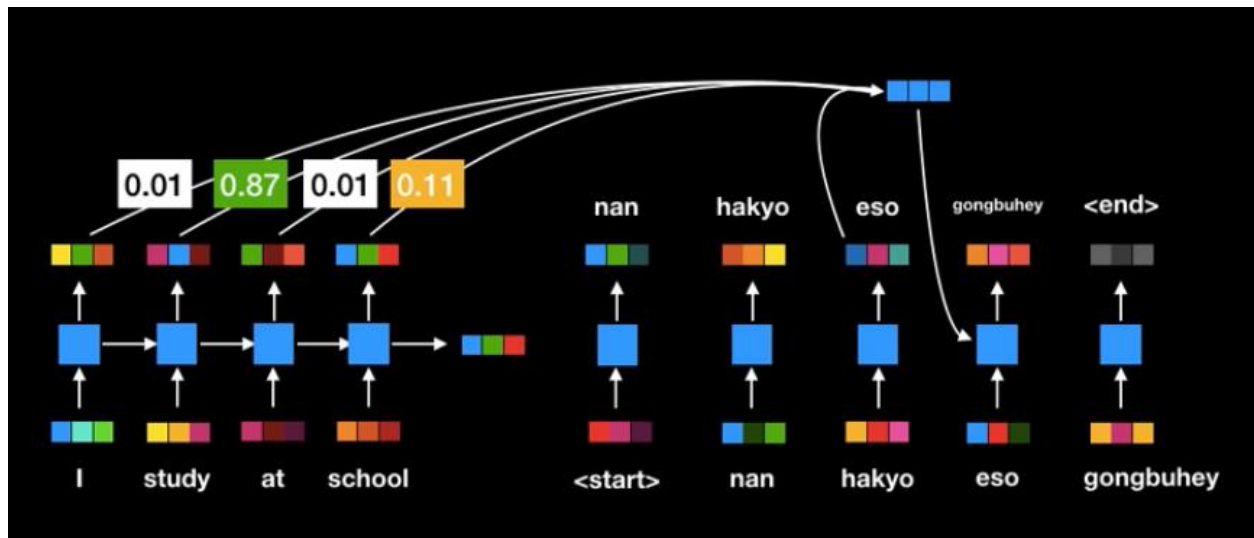


- Attention for speech to text (STT)



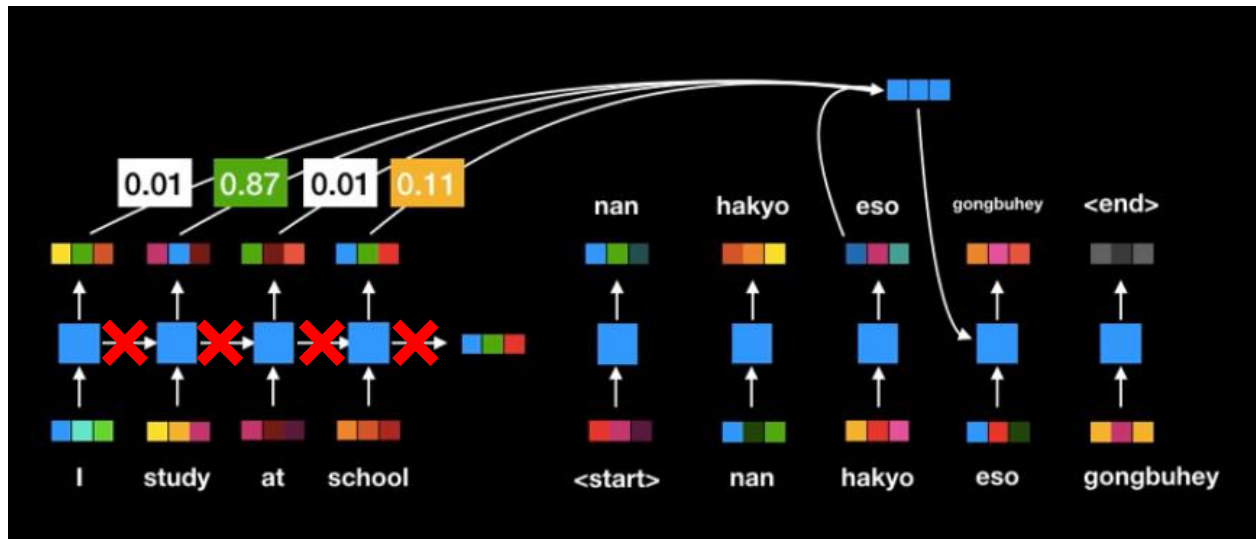
Attention 모델

- 문맥에 따라 동적으로 할당되는 encode의 Attention weight로 인한 dynamic context vector를 획득
- 기존 Seq2Seq의 encoder, decoder 성능을 비약적으로 향상시킴



Attention 모델

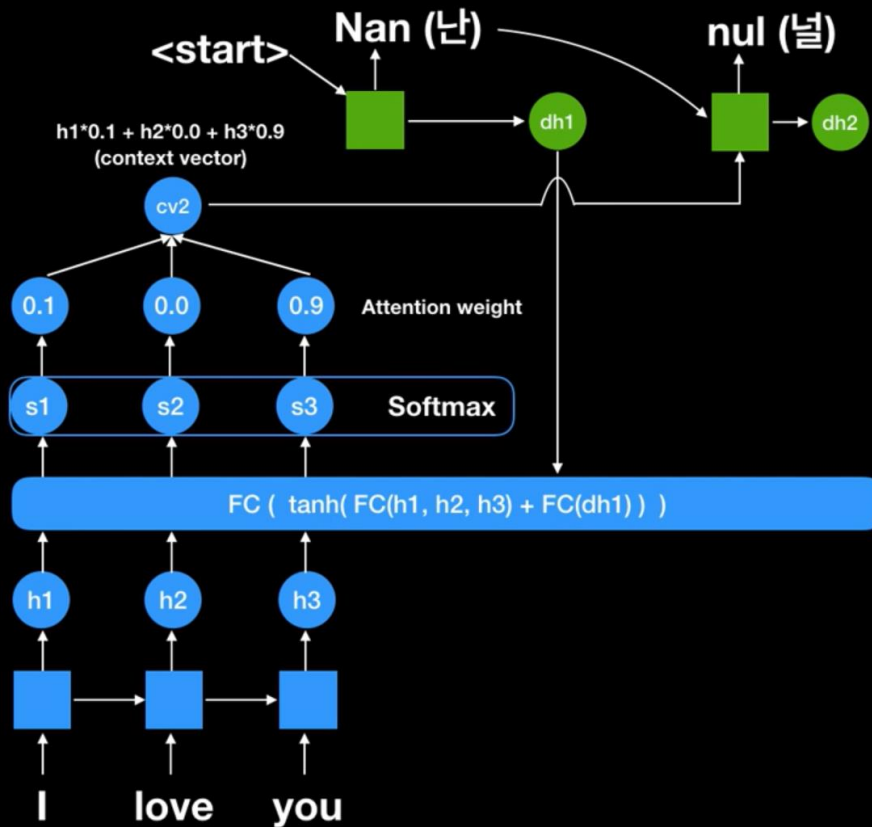
- 문맥에 따라 동적으로 할당되는 encode의 Attention weight로 인한 dynamic context vector를 획득
- 기존 Seq2Seq의 encoder, decoder 성능을 비약적으로 향상시킴
- 하지만, 여전히 RNN이 순차적으로 연산이 이뤄짐에 따라 연산 속도가 느림



그냥 **RNN**을 없애는건 어떨까?

Self-attention 모델

- Attention is all you need!
- RNN을 encoder와 decoder에서 제거

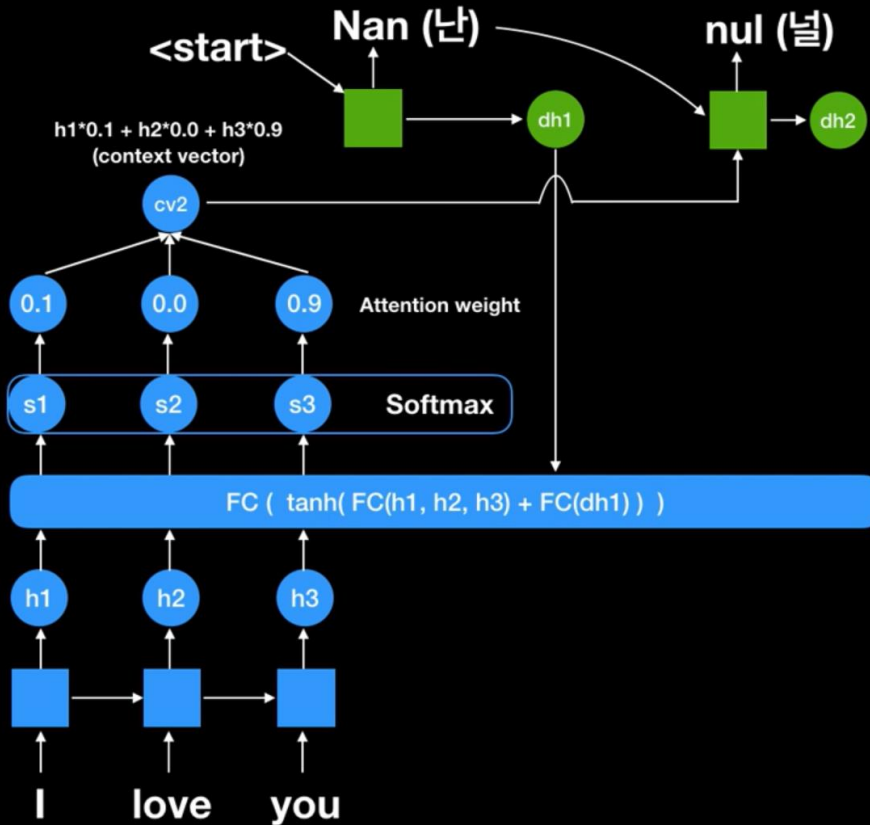


학습

Decoder 결과가 정답과 많이 다르다
 좋지 못한 context vector → 좋지 못한 attention weight
 Fully connected feed forward network에서 score를 조정
 s_1 - s_3 가 조정됨에 따라 Attention weight도 조정

Self-attention 모델

- Attention is all you need!
- RNN을 encoder와 decoder에서 제거



학습

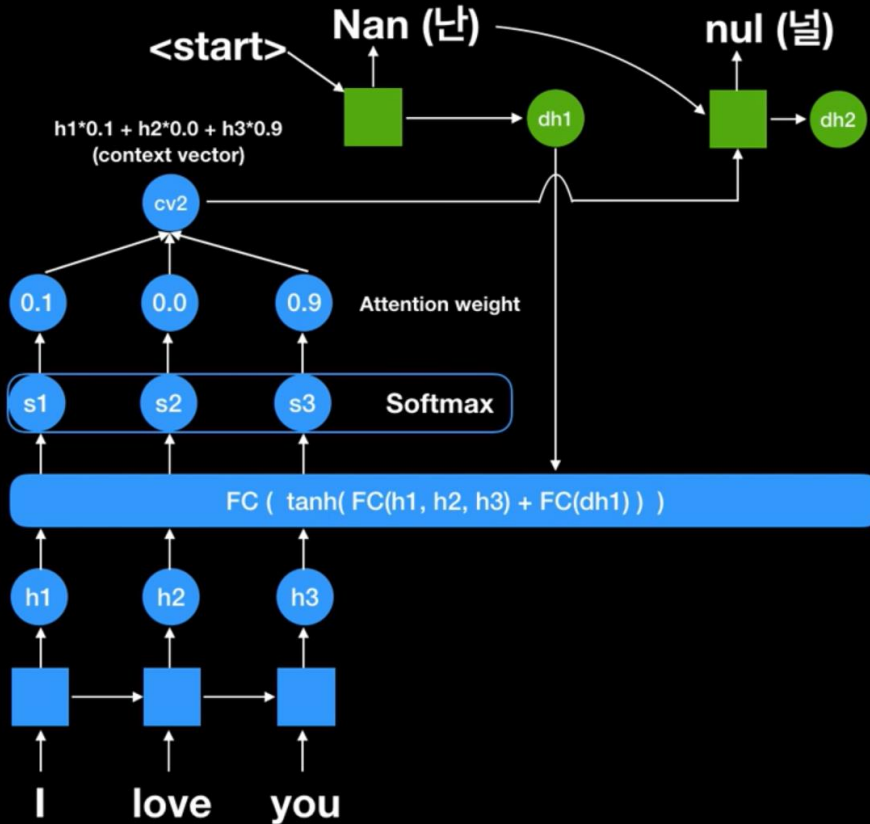
Decoder 결과가 정답과 많이 다르다
 좋지 못한 context vector → 좋지 못한 attention weight
 Fully connected feed forward network에서 score를 조정
 s_1-s_3 가 조정됨에 따라 Attention weight도 조정



RNN + attention에 적용된 attention은 decoder가 해석하기에 가장 적합한 weight를 찾고자 노력

Self-attention 모델

- Attention is all you need!
- RNN을 encoder와 decoder에서 제거



학습

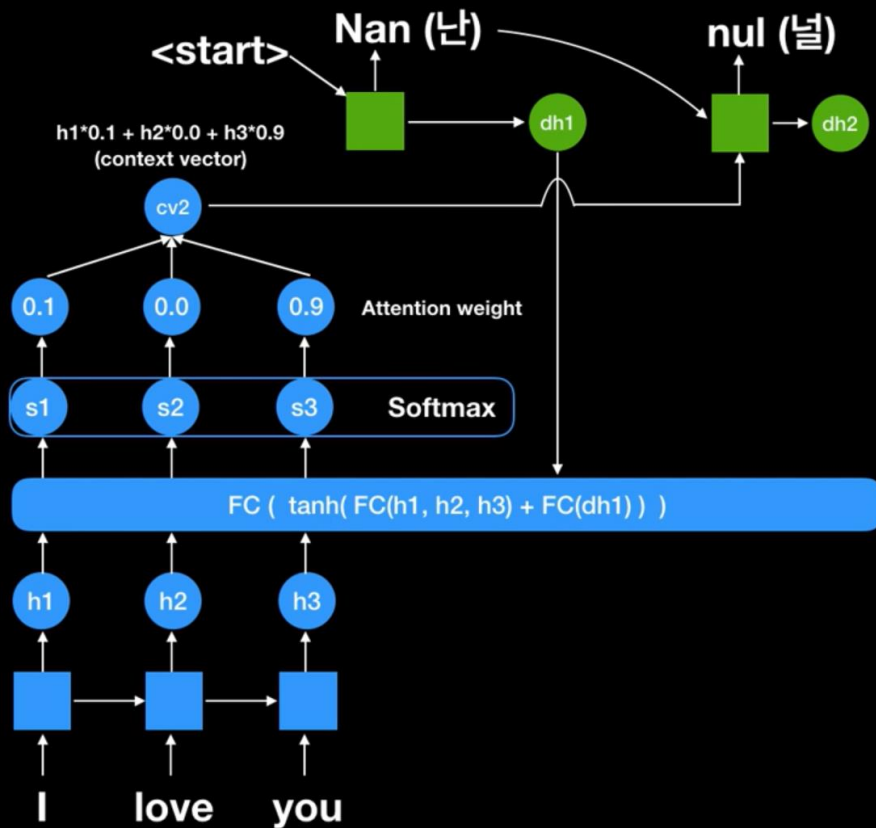
Decoder 결과가 정답과 많이 다르다
 좋지 못한 context vector \rightarrow 좋지 못한 attention weight
 Fully connected feed forward network에서 score를 조정
 s_1 - s_3 가 조정됨에 따라 Attention weight도 조정

RNN + attention에 적용된 attention은 decoder가 해석하기에 가장 적합한 weight를 찾고자 노력

Attention이 decoder가 아니라, input인 값을 가장 잘 표현할 수 있도록 학습하면?
 \rightarrow 자기 자신을 가장 잘 표현할 수 있는 좋은 embedding

Self-attention 모델

- Attention is all you need!
- RNN을 encoder와 decoder에서 제거



학습

Decoder 결과가 정답과 많이 다르다
 좋지 못한 context vector → 좋지 못한 attention weight
 Fully connected feed forward network에서 score를 조정
 s_1-s_3 가 조정됨에 따라 Attention weight도 조정

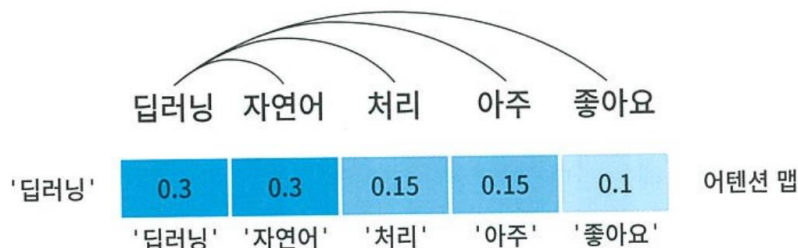
RNN + attention에 적용된 attention은 decoder가 해석하기에 가장 적합한 weight를 찾고자 노력

Attention이 decoder가 아니라, input인 값을 가장 잘 표현할 수 있도록 학습하면?
 → 자기 자신을 가장 잘 표현할 수 있는 좋은 embedding

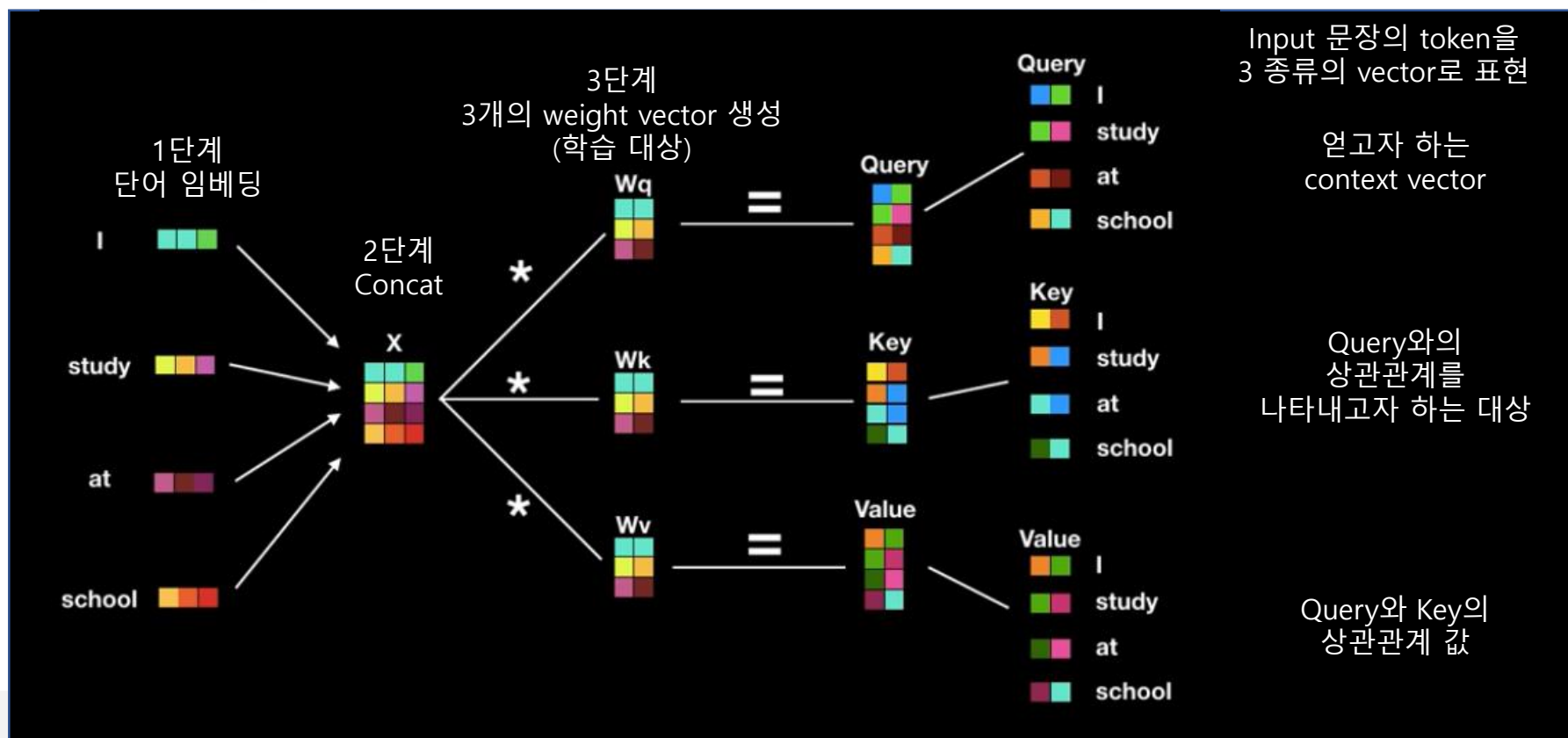
Self-attention 모델의 탄생!

Self-attention 모델

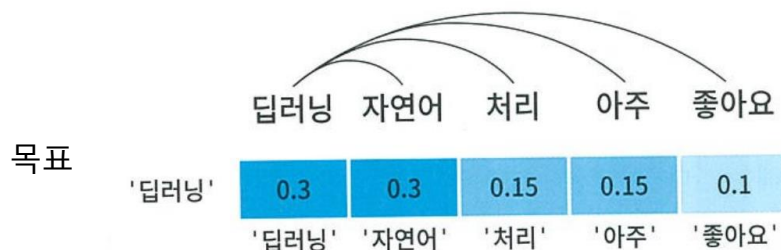
목표



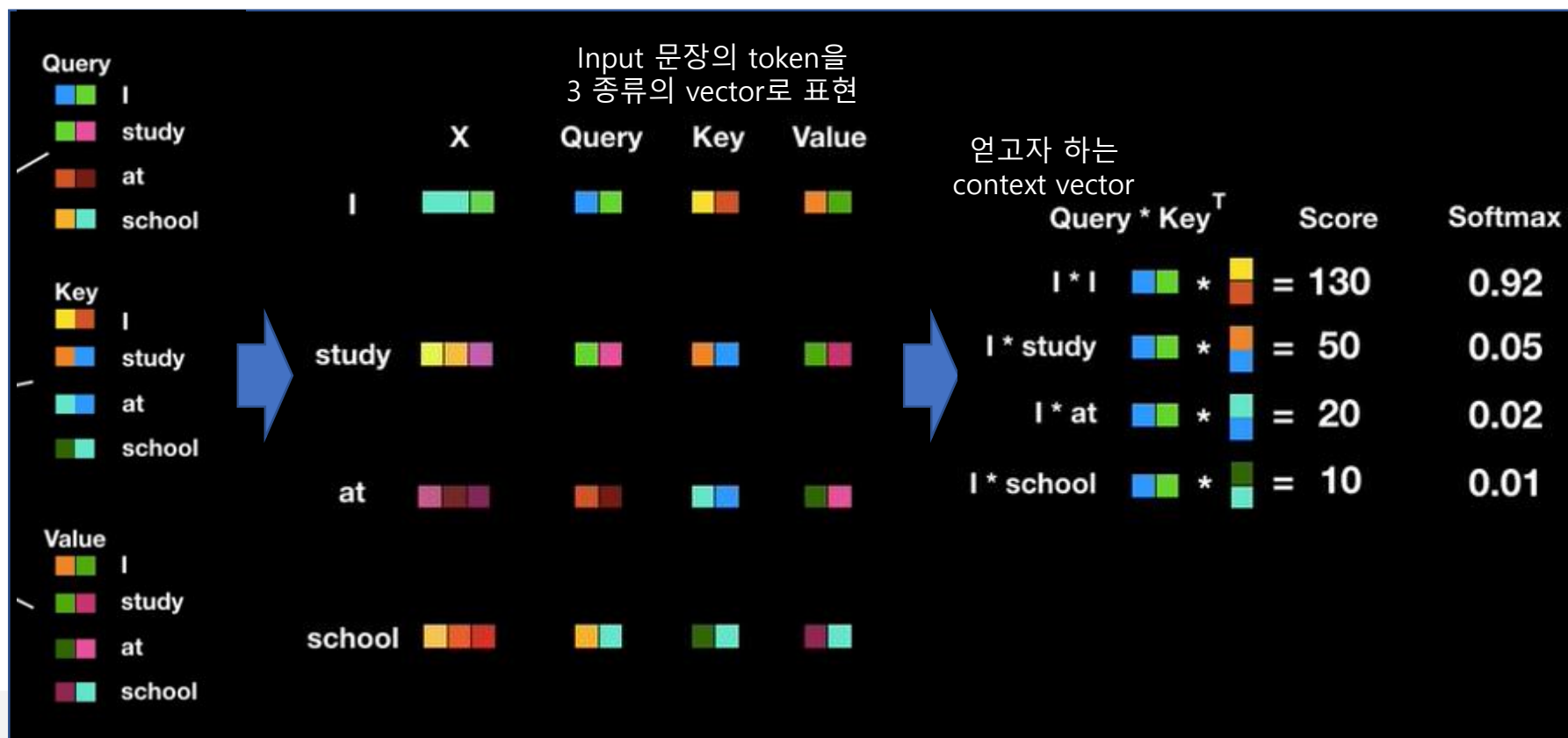
자기 자신을 잘 표현하는 attention을 구하자!



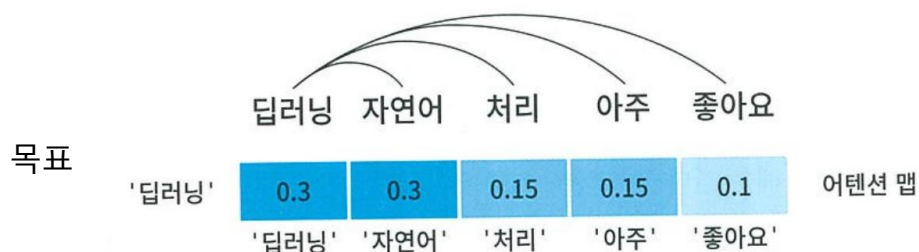
Self-attention 모델



자기 자신을 잘 표현하는 attention을 구하자!









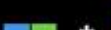




Self-attention 모델























































자기 자신을 잘 표현하는 attention을 구하자!

얻고자 하는
context vector

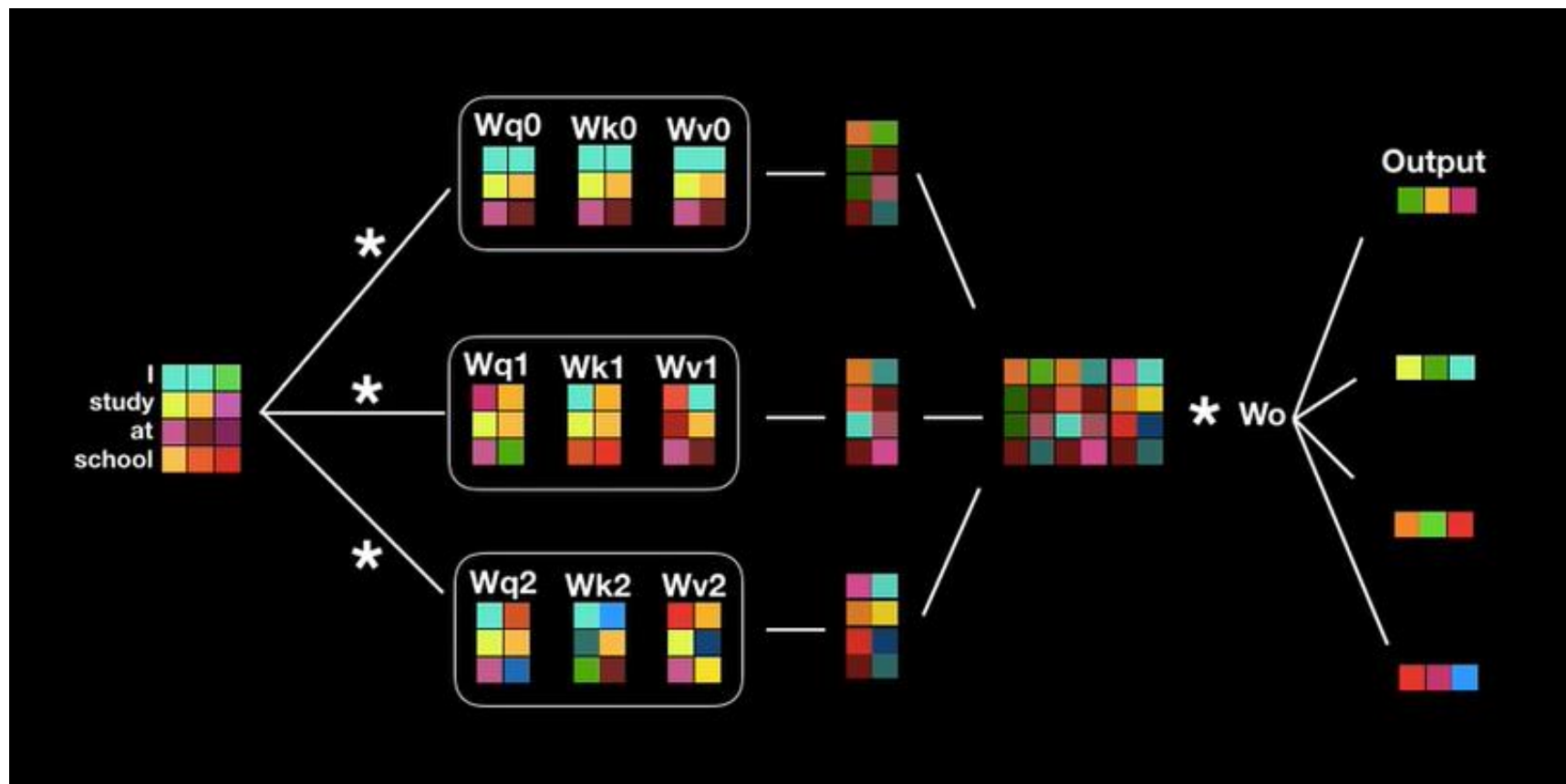
	Query * Key ^T	Score	Softmax	Value	Softmax * Value	Σ Softmax * Value (Attention layer output)
I * I	 *  = 130	0.92		I		 문장 속에서 단어 'I'가 지닌 의미!
I * study	 *  = 50	0.05		study		
I * at	 *  = 20	0.02		at		
I * school	 *  = 10	0.01		school		

Self-attention 모델

	Query * Key ^T	Score	Softmax	Value	Softmax * Value	Σ Softmax * Value (Attention layer output)
I	I * I  *  = 130	0.92		I	}	
	I * study  *  = 50	0.05		study		
	I * at  *  = 20	0.02		at		
	I * school  *  = 10	0.01		school		
study	study * I  *  = 30	0.02		}	}	
	study * study  *  = 110	0.70				
	study * at  *  = 20	0.03				
	study * school  *  = 70	0.25				
at	at * I  *  = 30	0.03		}	}	
	at * study  *  = 50	0.10				
	at * at  *  = 90	0.80				
	at * school  *  = 40	0.07				
school	school * I  *  = 30	0.01		}	}	
	school * study  *  = 80	0.27				
	school * at  *  = 23	0.02				
	school * school  *  = 160	0.70				

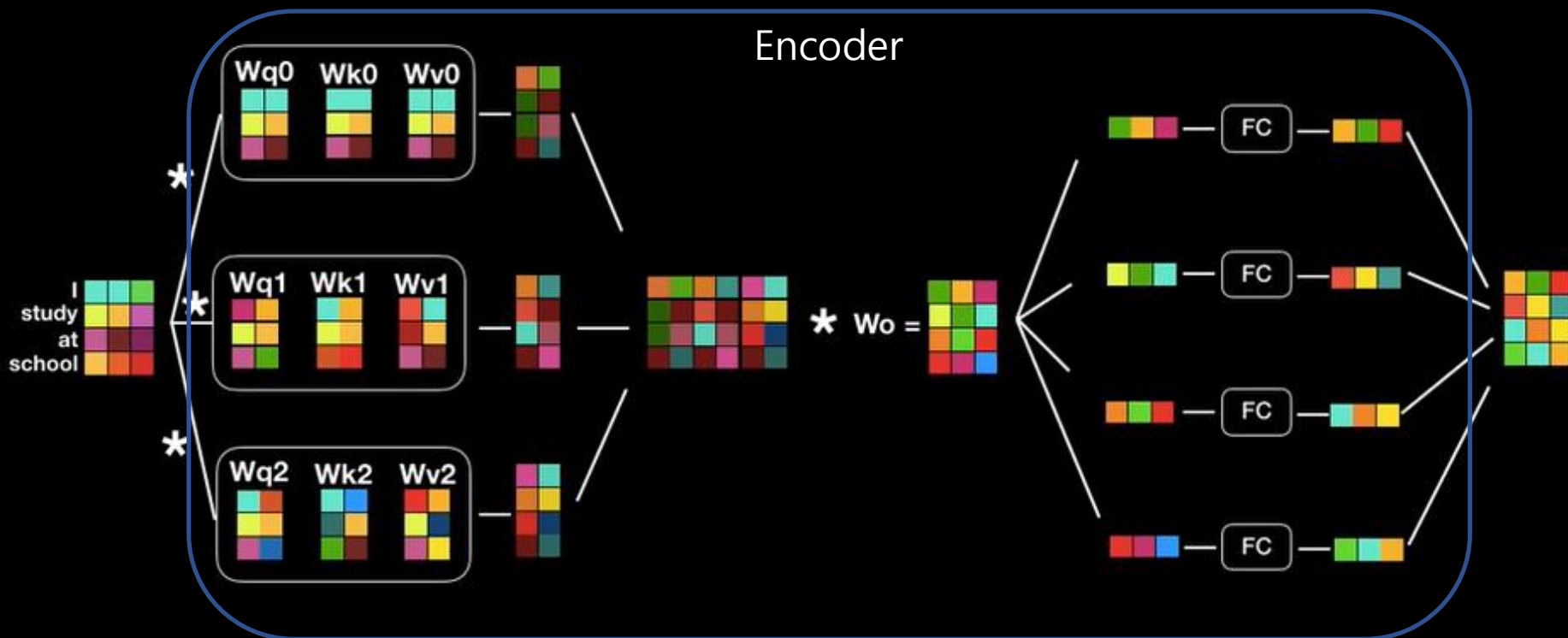
Multi-head Self Attention 모델

- Query, Key, Value로 구성된 attention layer를 동시에 여러 개를 수행



Multi-head Self Attention encoder 모델

- Query, Key, Value로 구성된 attention layer를 동시에 여러 개를 수행
- 최종적으로 자기 자신을 표현하는 vector 획득



Transformer 모델

- Transformer 모델은 multi-head attention으로 이루어진 encoder를 여러 층 쌓아서 encoding을 수행

