

유런 16주차

1. Natural language processing (자연어 처리)

- 학문 분야 (acl, emnlp, naacl)
- Low-level parsing(의미 단위의 low level) : Tokenization, stemming(studying, studied 는 다르지만 컴퓨터는 같다고 인식해야 함, 어근 추출)
- Word and phrase level : NER(Named Entity Recognition, 단일 단어나 여러 단어로 이루어진 고유 명사를 인식 ex. Newyork Times), POS(Part-Of-Speech, 문장 내에서 품사를 알려준다) tagging
- Sentence level : 감성 분류(Sentiment Analysis ex. this movie is not bad 를 긍정으로 인식할 수 있어야한다), 기계 번역(Machine Translation 한글 단어에 대한 번역과 어순을 고려해야 함)
- Multi-sentence and paragraph level : 논리적 내포 및 모순관계 예측(Entailment Prediction ex. 어제 좋으니 결혼했다, 어제 최소한 한명이 결혼했다-> 참 어제 한 명도 결혼하지 않았다->거짓), 독해기반 질의응답(question answering 원래는 키워드 검색을 했다면, 이제는 문맥을 알고 정확하게 알려준다), 챗봇(dialog systems 대화를 수행할 수 있는 기술), 요약(summarization 한 줄 요약)

2. Text mining (텍스트 마이닝) ⇒ 빅데이터 분야

- 주요 학회 : KDD, The WebConf(前 WWW), WSDM, CIKM, ICWSM
- 학문 분야
 - Extract useful information and insights from text and document data (토픽들을 분석)
 - 문서 군집화(Document clustering) ex) 토픽 모델링
 - Highly related to computational social science : 빅데이터를 기반으로 한 통계적으로 사회과학적 인사이트 산출 ex. 어떤 신조어를 많이 쓰고 이건 어떠한 사회 현상을 기반으로 하는지

3. Information retrieval (정보 검색) 구글이나 네이버에서의 검색 기능을 연구함, 이미 많이 발전했기에 발전 속도가 느림

- 주요 학회 : SIGIR, WSDM, CIKM, Recsys
- 학문 분야

- Highly related to computational social science
- 정보 검색 분야, 추천 시스템 (ex. 개인화 된 노래 추천, 영상 추천)

발전 과정

- 자연어 처리 분야는 컴퓨터 비전 혹은 영상처리 분야보다 늦게 발전하고 있지만 인공지능과 딥러닝 기술이 가장 활발히 적용되며 꾸준비 발전하는 분야 중 하나이다. 기존 머신러닝과 딥러닝 기술로 자연어 처리 문제를 해결하기 위해서는 주어진 텍스트 데이터를 숫자로 변환하는 '워드 임베딩(Word Embedding)' 과정을 거치게 된다.
- 텍스트 데이터는 문장을 구성하는 **순서** 정보가 중요하기 때문에 이를 받아들일 수 있는 특화 모델에 대한 연구가 필요했고, 그 대표적인 예로는 'RNN(**Recurrent Neural Network**)'(rule based 기계 번역보다 더 좋은 성능을 냄)이 있다. 이후 단점을 보완한 LSTM, GRU(gated recurrent neural network) 모델이 나와 사용되었다.
- 2017년에는 구글에서 발표한 'Attention is all YOU need' 라는 제목의 논문이 나오면서 '셀프 어텐션(Self-Attention)' 구조를 가진 '**트랜스포머(Transformer) 모델**'이 각광받기 시작했다. 최근 발표된 대부분의 모델들은 트랜스포머 모델을 기반으로 하는 것이 많으며, 트랜스포머 모델은 주로 사용되던 '기계 번역' 분야를 넘어 현재는 영상/신약 개발/시계열 예측 등에서도 다양하게 사용되고 있다.
- 최근에는 자가지도 학습(self-supervised Learning ex) i love study 에서 study 를 가리고 학습)이 가능한 BERT, GPT(모델 학습하는데 엄청난 전기가 요구됨.. 트랜스포머는 토큰이 하나 늘어나면 quadratic 하게 계산량이 늘어나기 때문) 와 같은 모델의 유행하고 있다.

각각의 단어를 벡터 공간의 한 점으로 나타내는 워드 임베딩에 대해서 알아보자.

Word Embedding 이란?

'워드 임베딩'은 각 단어를 좌표공간에 최적의 벡터로 표현하는(임베딩하는) 기법을 말한다.

그렇다면 표현된 벡터값이 '최적'인지를 어떻게 알 수 있을까?

- kitty : 아기 고양이
- cat : 고양이
- hamburger : 햄버거

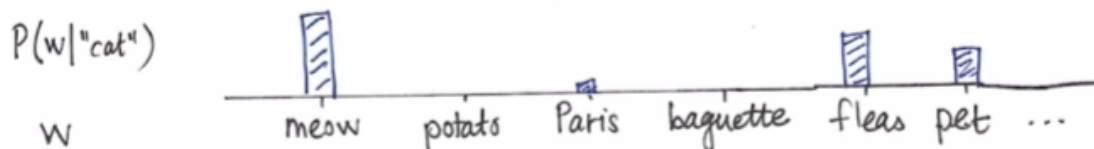
위 단어들을 벡터를 통해 좌표공간으로 표현한다면, 'kitty'와 'cat'은 비슷한 위치할 것이다. 그러나 'hamburder'는 꽤 먼 거리에 표현될 것이다. 이와 같이 유사한 단어는 가까이, 유사하지 않은 단어는 멀리 위치하는 것을 '최적의 좌표값'으로 표현할 수 있다.

또 다른 예로 감정을 분류를 한다고 했을 때,

- "기쁨", "환희"는 긍정적인 감정을 나타내는 단어들과 함께
- "분노", "증오"는 부정적인 감정을 나타내는 단어들과 비슷한 위치에 맵핑이 될 것이다.

Word2Vec Idea

- "You shall know a word by the company it keeps" –J.R. Firth 1957
- Suppose we read the word "cat"
 - What is the probability $P(\underline{w}|\text{cat})$ that we'll read the word \underline{w} nearby?



- Distributional Hypothesis: The meaning of "cat" is captured by the probability distribution $P(\underline{w}|\text{cat})$
- '워드 투 벡터'의 아이디어는 "문장 내에서 비슷한 위치에 등장하는 단어는 유사한 의미를 가질 것이다" 에서 출발합니다. 즉, 주변에 등장하는 단어들을 통해 중심 단어의 의미가 표현될 수 있다는 것으로 추정을 시작합니다. (ex. the cat purrs, this cat hunts mice \Rightarrow the 는 cat 을 꾸며주는구나, purr 란 hunt 는 cat 이 하는 행위구나, mice 는 hunt 의 대상이구나)

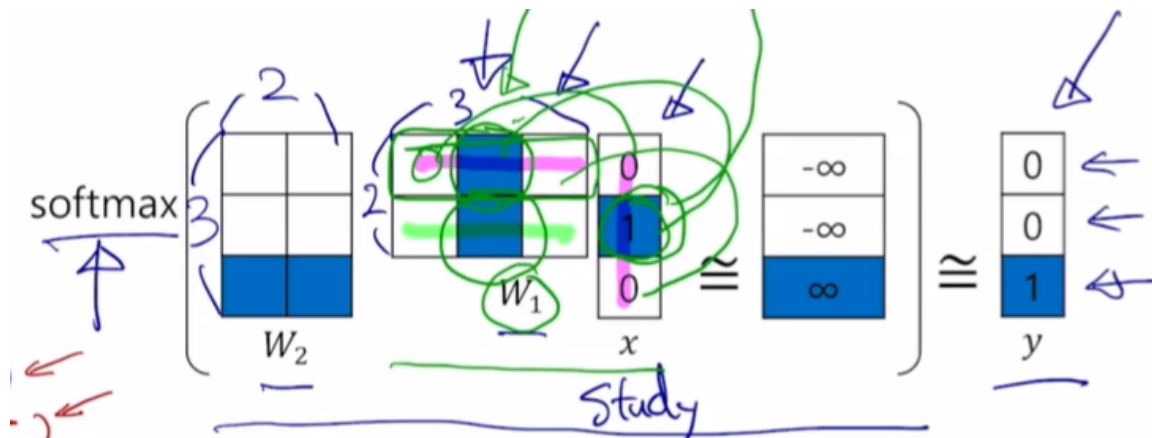
How Word2Vec Algorithm Works

Word Embedding: Word2Vec, GloVe



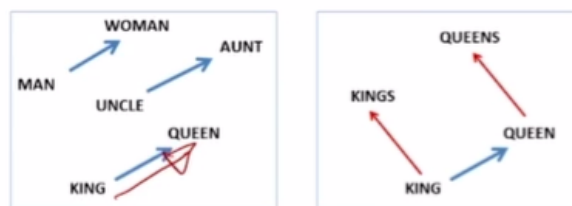
- Sentence : "I study math."
- Vocabulary: {"I", "study", "math"}
- Input: "study" [0, 1, 0]
- Output: "math" [0, 0, 1]
- Columns of W_1 and rows of W_2 represent each word
- E.g., 'study' vector : 2nd column in W_1 , 'math' vector : 3rd row in W_2 .
- The 'study' vector in W_1 and the 'math' vector in W_2 should have a high inner-product value.

슬라이딩 윈도우로 단어 쌍을 만든다.



- 이를 위해 우선 워드를 토크나이징(Tokenizing)해준 후, 유니크한 단어만 모아서 사전 (Vocabulary)을 만들어주어야 한다. 그 이후 문장에서 중심단어를 위주로 학습 데이터를 구축해준다.
- 예를 들어 "I study math"라는 문장의 중심단어가 study 라고 한다면, (I study), (study I) , (study math) 와 같은 단어쌍을 학습 데이터로 구축한다.
- 보통 input vector 를 word embedding vector 로 사용한다.
- * 토크나이징(Tokenizing)이란? : 말그대로 문자(Text)를 컴퓨터가 이해할 수 있는 Token이라는 숫자 형태로 바꿔주는 행위

- 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.,
- $\text{vec}[\text{queen}] - \text{vec}[\text{king}] = \text{vec}[\text{woman}] - \text{vec}[\text{man}]$

Word2Vec의 특성

- 워드투벡터를 통해 단어를 임베딩하면 queen - king 그리고 woman - man , 마지막으로 aunt - uncle 의 벡터가 비슷한 것을 볼 수 있다. 해당 결과가 의미하는 것은 여성과 남성의 관계성을 잘 학습했다는 것을 의미한다.

버락_오바마-미국+스 타워즈	아나킨/Noun_스카이 워커/Noun	-
아카라카-연세대학교 +고려대학교	입실렌티/Noun	입실렌티/Noun
아이폰-휴대폰+노트 북	아이패드/Noun	아이패드/Noun
컴퓨터공학-자연과학 +인문학	법학/Noun	게임학/Noun

Intrusion detection

- Word intrusion detection
 - staple hammer saw drill
 - math shopping reading science
 - rain snow sleet sun
 - eight six seven five three owe nine
 - breakfast cereal dinner lunch
 - england spain france italy greece germany portugal australia

staple 과 나머지 세 개 벡터의 평균 값과 평균 거리를 구한다.

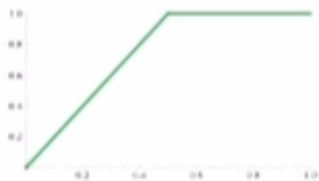
Application of Word2Vec

- Word2Vec은 그 자체로도 의미가 있지만, 뿐만 아니라 다양한 테스트에서 사용되고 있다.
- Machine translation : 단어 유사도를 학습하여 번역 성능을 더 높여준다.
- Sentiment analysis : 감정분석, 긍부정분류를 돕는다.
- Image Captioning : 이미지의 특성을 추출해 문장으로 표현하는 테스트를 돕는다.

Glove라는 또다른 Word Embedding 기법에 대해 알아보자.

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-occurent matrix.

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


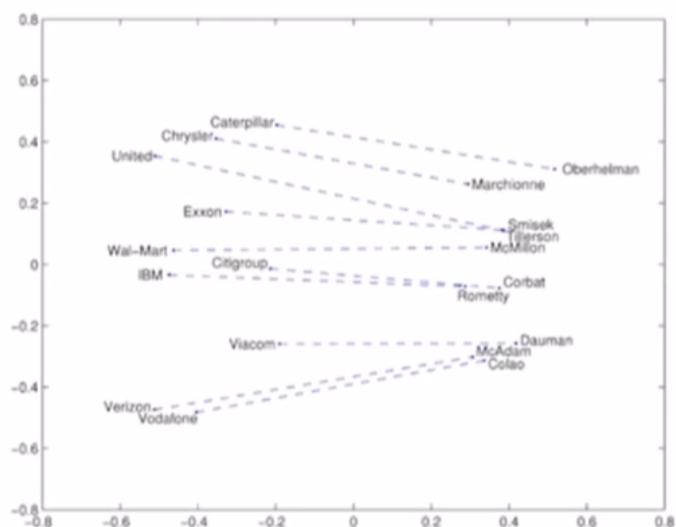
- Fast training
- Works well even with a small corpus

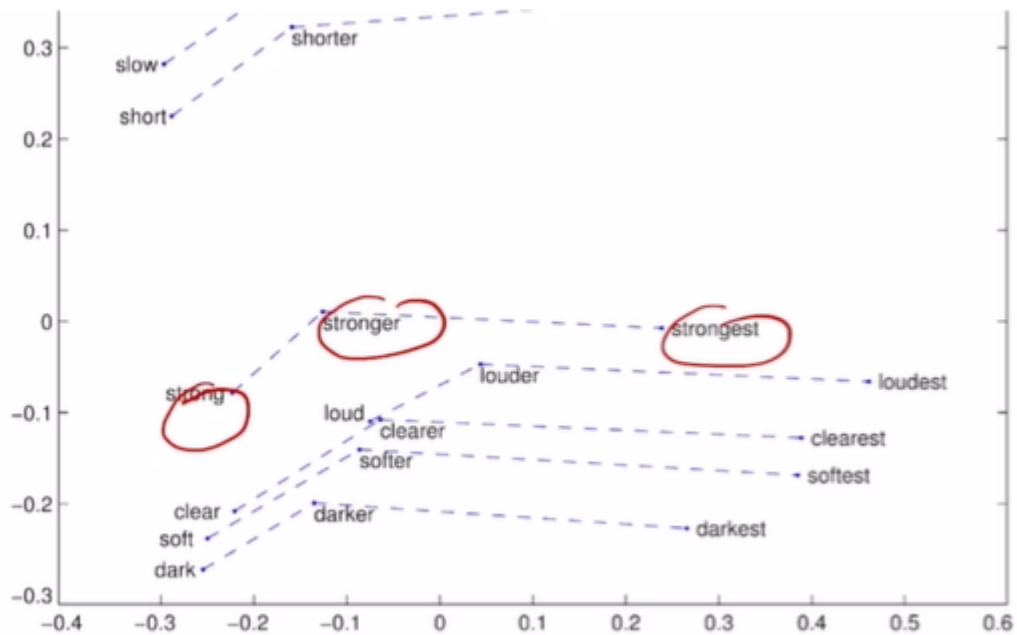
$P_{i,j}$ 는 한 윈도우 내에서 동시에 몇 번 나타났는가를 나타낸다(미리 계산해서 ground truth 로 사용, 따라서 학습이 빠르고 적은 데이터에 대해서도 잘 작동한다). 내적값이 이 값과 가까워지도록 설정한다.

Glove : Global Vectors for Word Representation

- Glove는 Word2Vec과 다르게 사전에 미리 각 단어들의 동시 등장 빈도수를 계산하며, 단어간의 내적값과 사전에 계산된 값의 차이를 줄여가는 형태로 학습한다.
- Word2Vec는 모든 연산을 반복하지만, Glove는 사전에 계산된 Ground Truth를 사용해 반복계산을 줄일 수 있다.
- 따라서 Word2Vec보다 더 빠르게 동작하며, 더 적은 데이터에서도 잘 동작한다.

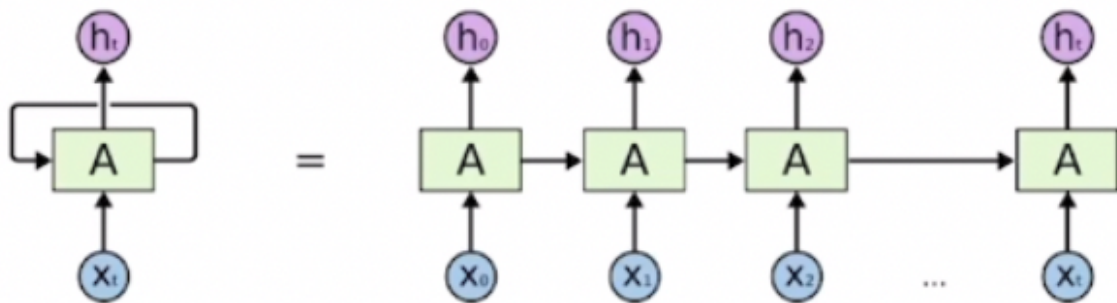
- company – ceo





비교급, 최상급에서도 잘 학습이 된 것을 알 수 있다.

RNN(Recurrent Neural Network)

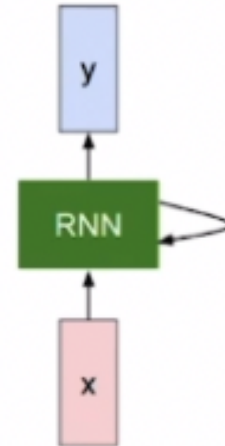


An unrolled recurrent neural network.

- RNN은 현재 타임스텝에 대해 이전 스텝까지의 정보를 기반으로 예측값을 산출하는 구조의 딥러닝 모델이다.
- 매 타임스텝마다 동일한 파라미터를 가진 모듈(A)을 사용하므로, '재귀적인 호출'의 특성을 보여주어 'Recurrent Neural Network'라는 이름을 가지게 되었다.
- 왼쪽, 오른쪽과 같이 표현할 수 있다.

RNN 계산 방법

$$h_t = f_W(h_{t-1}, x_t)$$



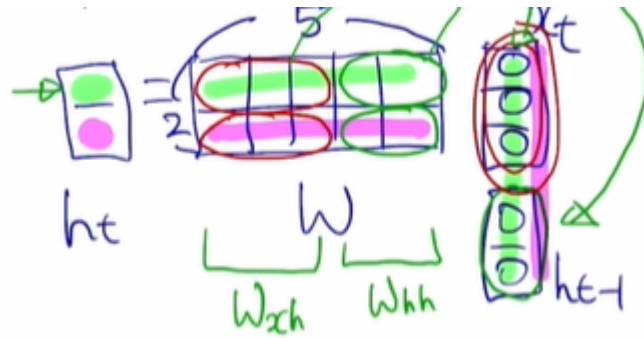
- t : 현재 타임스텝(time step) , w : 웨이트(weight)
- h_{t-1} : old hidden-state vector 입력
- x_t : input vector at some time step 입력
- h_t : new hidden-state vector 계산 결과
- f_w : RNN function with parameters W
- y_t : output vector at time step t h_t 를 기반으로 한 최종 결과

$$h_t = f_W(h_{t-1}, x_t)$$

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

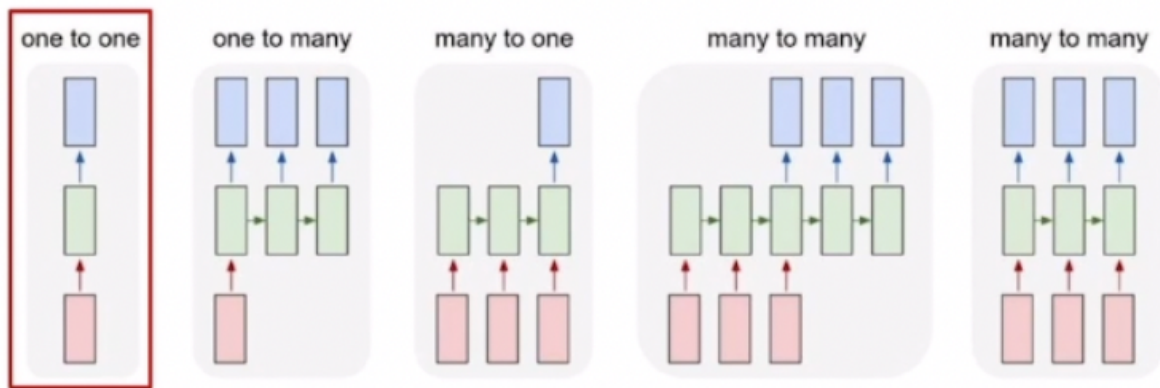
$$y_t = W_{hy}h_t$$

$W_{h_t \rightarrow y_t}$



매 타임스텝마다 파라미터는 변하지 않는다.

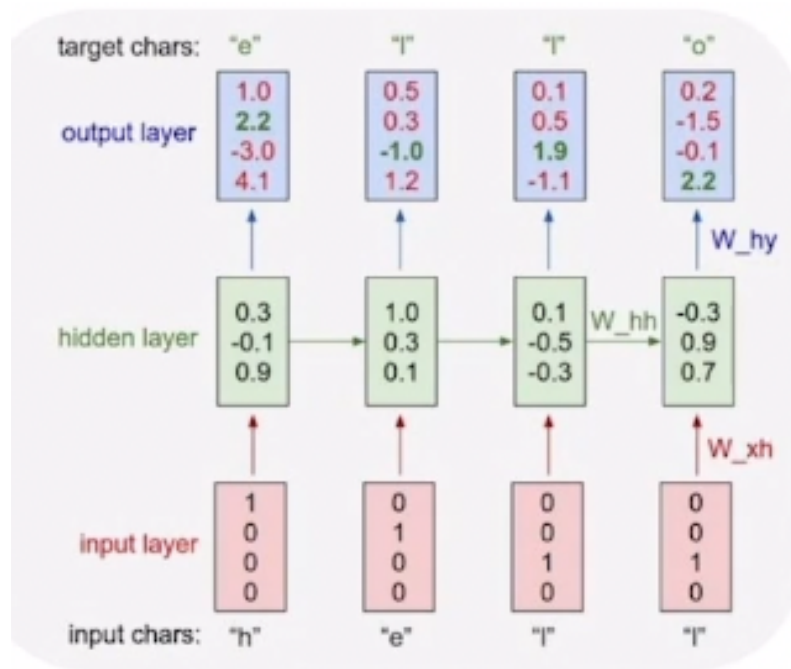
다양한 타입의 RNN 모델



1	one to one(not sequence)	[키, 몸무게, 나이]와 같은 정보를 입력값으로 할 때, 이를 통해 저혈압/고혈압인지 분류하는 형태의 태스크
2	one to many(입력: 1 time step출력: multi time step)	'이미지 캡셔닝'과 같이 하나의 이미지를 입력값으로 주면 설명글을 생성하는 태스크
3	many to one	감성 분석과 같이 문장을 넣으면 긍정/부정 중 하나의 레이블로 분류하는 태스크
4	many to many	기계 번역과 같이 입력값을 끝까지 다 읽은 후, 번역된 문장을 출력해주는 태스크
5	many to many	실시간성이 요구되는 비디오 분류와 같이 영상의 프레임 레벨에서 예측하는 태스크 혹은 각 단어의 품사에 대해 태깅하는 POS와 같은 태스크

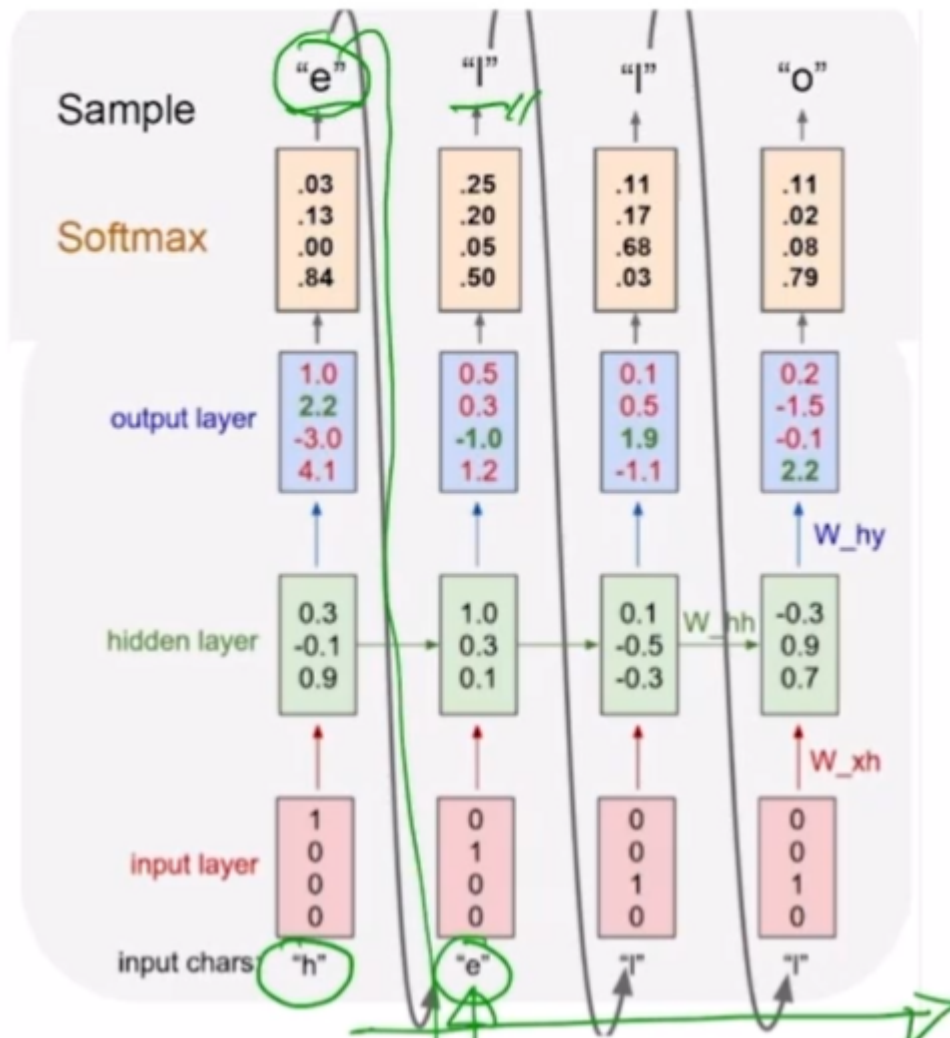
character-level Language Model

many to many



vocabulary : [h,e,l,o] 이므로 각각을 4 dimension 의 1 hot vector 로 나타낼 수 있다.

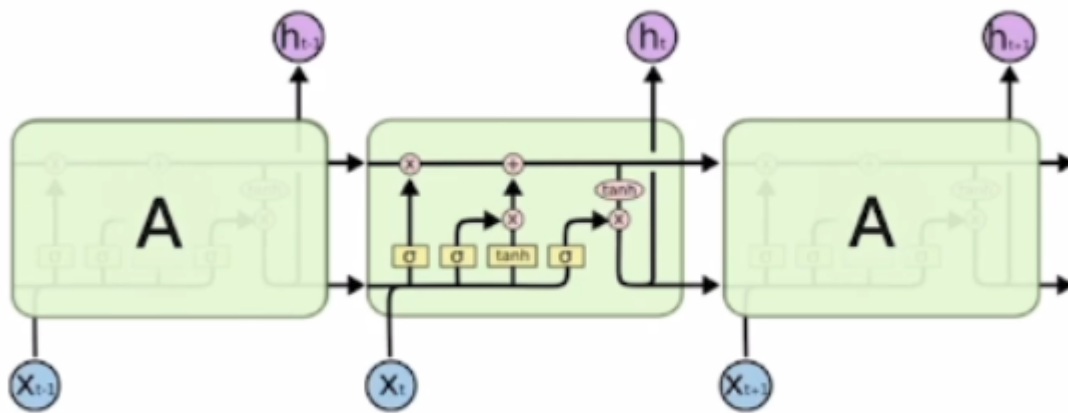
- 언어 모델이란 이전에 등장한 문자열을 기반으로 다음 단어를 예측하는 태스크를 말한다. 그중에서도 캐릭터 레벨 언어 모델(character-level Language Model)은 문자 단위로 다음에 올 문자를 예측하는 언어 모델이다.
- 예를 들어, 그림과 같이 맨 처음에 "h"가 주어지면 "e"를 예측하고, "e"가 주어지면 "l"을 예측하고, "l"이 주어지면 다음 "l"을 예측하도록 hidden state가 학습돼야 한다.
- 이때 각 타임스텝별로 output layer를 통해 차원이 4(유니크한 문자의 개수) 벡터를 출력해주는데 이를 logit이라고 부르며, softmax layer를 통과시키면 원-핫 벡터 형태의 출력값이 나오게 된다. 초록색 숫자가 ground truth 이고 이것과 최대한 가까워지게 네트워크를 학습시킨다.



RNN에서 발생하는 Long-Term-Dependency 현상에 대해 학습했는데 길이가 길어질수록 학습이 잘 이뤄지지 않는다는 문제점이 있다. 이를 해결하기 위해 LSTM(Long Short-Term Memory)라는 모델이 나오게 된다.

LSTM 이 RNN 의 Long-Term-Dependency 를 어떻게 극복하는지 이해해보자.

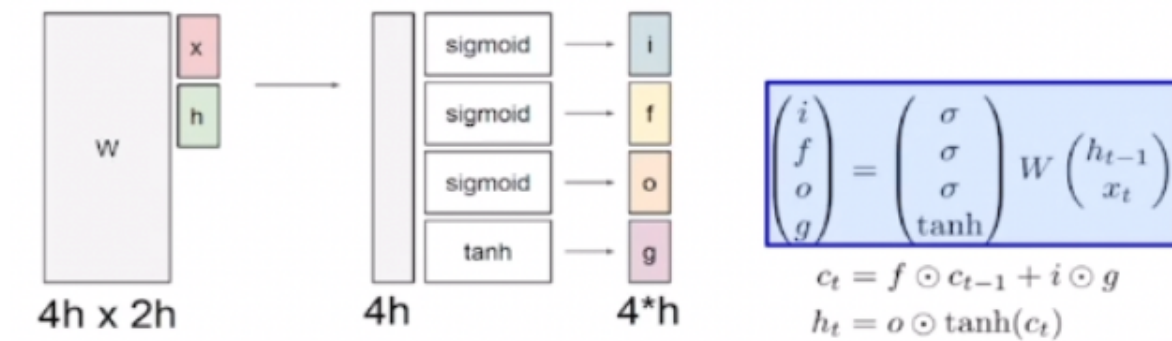
LSTM : Long Short-Term Memory



The repeating module in an LSTM contains four interacting layers.

- LSTM의 중심 아이디어는 단기 기억으로 저장하여 이걸 때에 따라 꺼내 사용함으로 더 오래 기억할 수 있도록 개선하는 것이다. 다시 말해 Cell state에는 핵심 정보들을 모두 담아두고, 필요할 때마다 Hidden state를 가공해 time step에 필요한 정보만 노출하는 형태로 정보가 전파된다.

LSTM의 여러 gate 설명



- 위 그림을 살펴보면 input으로 x_t, h_{t-1} 이 들어오게 되고 이를 W에 곱해준 후 각 sigmoid or tanh로 연산해준다. 게이트별로 설명은 아래와 같습니다. Ifog 로 불리는 게이트들은 그림에서 순서대로 나타난다.

- I : Input gate로 불리며, cell 에 쓸 지말지를 결정하는 게이트이다. 즉, 들어오는 input 에 대해서 마지막에 sigmoid를 거쳐 0-1 사이 값으로 표현해준다. 이 값은 cell state와 hidden state 두 갈래로 흐르게 된다.
 - 표현식 : $\text{sigmoid}(W(x_t, h_{t-1}))$

x_t

h_{t-1}

- f : Forget gate 로 불리며, 정보를 어느정도로 지울지를 0~1사이의 값으로 나타낸다.
 - 표현식 : $\text{sigmoid}(W(x_t, h_{t-1}))$

x_t

h_{t-1}

- o : Output gate로 불리며, Cell 정보를 어느정도 hidden state에서 사용해야할 지를 0~1사이 값으로 나타낸다.
 - 표현식 : $\text{sigmoid}(W(x_t, h_{t-1}))$

x_t

h_{t-1}

- g : Gate gate로 불리며, 어느정도로 Cell state에 반영해야할 지를 -1 ~ 1 사이의 값으로 나타낸다.
 - 표현식 : $\tanh(W(x_t, h_{t-1}))$

x_t

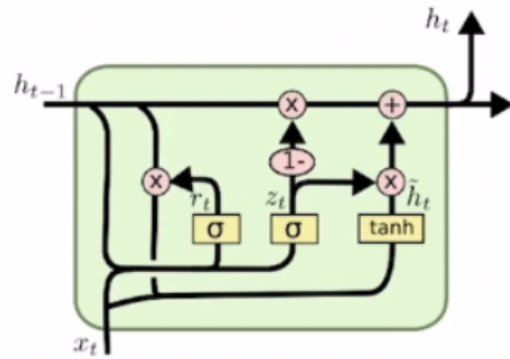
h_{t-1}

LSTM이 RNN과 다른점

- LSTM의 특징은 각 time step마다 필요한 정보를 단기 기억으로 hidden state에 저장하여 관리되도록 학습하는 것이다.
- 오차역전파(backpropagation) 진행시 가중치(W)를 계속해서 곱해주는 연산이 아니라, forget gate를 거친 값에 대해 필요로하는 정보를 덧셈을 통해 연산하여 그래디언트 소실/증폭 문제를 방지한다.

GRU : Gated Recurrent Unit

- $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
- $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
- $\tilde{h}_t = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t])$
- $h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$
- c.f) $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$
in LSTM



- LSTM과 전체적인 동작원리는 유사하지만, Cell state, Hidden state를 일원화하여 경량화한 모델이다.
- GRU에서 사용되는 h_{t-1} 은 LSTM에서의 c_t 와 비슷한 역할을 한다.
- forget gate 대신 (1-input gate)를 사용하여 h_t 를 구할때 가중평균의 형태로 계산하게 된다.
- 계산량과 메모리 요구량을 LSTM에 비해 줄여준 모델이면서 동시에 성능면에서도 LSTM과 비슷하거나 더 좋은 성능을 내는 모델이다.

RNN , LSTM , GRU 요약

- RNN은 들어오는 입력값에 대해서, 많은 유연성을 가지고 학습되는 딥러닝 모델이다.
- RNN에서는 그래디언트 소실/증폭 문제가 있어 실제로 많이 사용되지는 않지만, RNN 계열의 LSTM, GRU 모델은 현재도 많이 사용되고 있다.
- LSTM과 GRU 모델은 RNN과 달리 가중치를 곱셈이 아닌 덧셈을 통한 그래디언트 복사로 그래디언트 소실/증폭 문제를 해결했다.