Machine Learning

6장 텍스트자료 분석

고려대학교 통계학과
박유성

Contents

01 Word2Vec과 Glove

02 텍스트자료에 특화된 Word Embedding

03 Word Embedding 사례분석

텍스트자료 분석

- 한글을 영어로 번역할 때 단어의 순서는 언어번역에 중요한 역할을 한다. 또한 내일의 주식가격을 예측하고자 할 때 오늘의 주식가격이 일주일 전의 주식가격보다 좀 더 많은 비중을 주고 예측을 하게 된다.
- 이처럼 자료의 순서 및 시점이 중요한 역할을 할 때 적용되는 딥러닝이 RNN (recurrent neural networks)이다.
- RNN의 입력자료 형태는 3D 텐서이며 (표본수, 시간스탭수, 특성변수수)의 크기를 가지고 있다.
- 그러므로 하나의 표본은 m개의 시간스탭(time steps)과 p개의 특성변수로 구성된 2D텐서가 된다

텍스트자료 분석

- 텍스트자료에서 표본은 분석의 목적에 따라 문장이 될 수도 있고 문서도 될 수 있다. 이해를 쉽게 하기 위해 하나의 문장(sentence)이 하나의 표본이라고 가정하자.
- 각 문장을 구성하는 서로 다른 단어의 집합이 특성변수이다. 그러나 각 문장의 길이가 다르므로 특성변수의 수가 다르게 되며 이는 각 표본이 다른 크기의 특성변수를 가지게 되는 문제가 발생한다.
- 표본은 동일한 형태가 반복되어야만 문장 내의 단어들 간에 존재하는 통계적 패턴을 찾을 수 있다.
- 그러므로 텍스트 자료분석을 위해서 각 문장의 특성변수의 숫자는 동일해야 하며 시간스탭 크기도 동일해야 한다.

시계열자료 분석

- 주식가격을 예측하기 위해, 주식가격에 영향을 미치는 특성변수도 필요하지만 과거의 주식가격도 현재의 주식가격을 예측하는데 중요한 특성변수가 된다.
- 그러므로 현재의 주식가격을 설명하기 위해 얼마나 먼 과거의 주식가격이 필요한지에 따라 시간스탭의 크기가 결정된다.
- 그러나 좀 더 중요한 것은 (시간스탭수, 특성변수수)로 구성된 2D 텐서 표본이 n개 있을 때 n개의 표본이 동일한 분포를 가지고 있느냐?가 보장되어야 한다.
- 동일한 분포를 가지고 있어야 만 n개의 반복된 표본에 의해 주식가격의 통계적 패턴을 찾을 수 있기 때문이다.
- 동일한 분포를 갖기 위한 최소한의 조건은 주식가격이 정상성(stationary)을 만족해야 한다.

시계열자료 분석

- 정상성을 직관적으로 이해하기 위해 최근 3일의 주식가격을 알면 내일 주식가격을 예측할 수 있다고 가정하자.
- y_t를 예측하기 위해 $y_{t-1}, y_{t-2}, y_{t-3}$가 필요하다면, $(y_{t-1}, y_{t-2}, y_{t-3})$를 특성변수로 하고 y_t를 목적변수로 하는 딥러닝모형을 구축하면 된다.
- 시점을 임의로 $t = 15,10,5,20,55$으로 뽑았을 때
- $[(y_{14}, y_{13}, y_{12}), y_{15}]$, $[(y_9, y_8, y_7), y_{10}]$, $[(y_4, y_3, y_2), y_5]$, $[(y_{19}, y_{18}, y_{17}), y_{20}]$, $[(y_{54}, y_{53}, y_{52}), y_{55}]$를 이용하여
- 딥러닝모형의 모수를 추정하게 된다(물론, 실제 딥러닝을 적용하기 위해서는 훨씬 큰 표본이 필요하다).
- 딥러닝모형은 이 5개의 자료를 이용하여 연속된 4개 시점간의 선형 및 비선형 패턴을 구하는 문제이므로 이 5개 자료의 패턴은 서로 간에 비슷해야만 한다.

시계열자료 분석

- 통계적인 관점에서 살펴볼 때, 이 5개 자료는 숨겨진 동일한 패턴에 오차가 추가된 형태여야만 한다.
- 그러므로 5개 자료가, 예를 들어 평균이 다르거나(이는 평균값이 시점에 의존해서 시점이 증가하면서 증가하거나 감소하는 경향) 자기상관관계 autocorrelation)가 다르면 적용할 수 없다는 의미를 가지게 된다.
- 이러한 특성을 가진 시계열자료는 정상성을 위배하는 대표적인 예이다.
- 그러므로 딥러닝모형에 시계열자료를 적용할 때, 자료가 정상성인지를 점검하여야 하며
- 정상성이 아닐 경우 적절한 차분(differencing)을 통해 정상성자료로 변환시켜야 한다.

Word2Vec과 Glove

- SNS 등의 댓글에 특정 주제에 대해 찬성인지 반대인지를 분류하거나 신문의 기사가 사회, 경제, 정치에 관련된 내용인지를 분류하고, 특정소설의 문장과 구성을 통해 저자 및 출판년도를 맞추거나, 영어를 불어로 번역하는 것 등은 모두 텍스트 자료를 기반으로 하는 딥러닝 관심분야이다.
- 텍스트자료를 분석하기 위해서는 텍스트자료의 수량화가 필요하다.
- 특정주제에 대한 SNS 상에 나타난 댓글은 하나의 표본이 되고 신문기사의 경우 기사가 하나의 표본이 된다.
- 이러한 표본의 수량화는 표본을 구성하는 단어(word)나 문자(character)를 토큰화(tokenize)하고 토큰에 인덱스(index)를 부여한 후, 인덱스에 의미를 부여하는 과정으로 요약할 수 있다

Word2Vec과 Glove

- 가장 일반적인 토큰은 단어이며 전체표본에 나타난 모든 서로 다른 유일한 단어를 모아서 각각의 단어에 1:1로 대응되는 인덱스를 부여한다.
- 예를 들어 “The cat sits on the mat”, “The other cat runs over the mats”라는 두 개의 문장이 있을 때
- 각 문장은 1개의 표본이 되고 전체표본은 2개인 간단한 예제이다.
- 전체표본을 구성하는 유일한 단어는 ‘the’, ‘cat’, ‘sits’, ‘on’, ‘mat’, ‘other’, ‘runs’, ‘over’, ‘mats’이다.
- 이러한 단어의 분리가 토큰을 단어로 하는 토큰화이다.

Word2Vec과 Glove

- ‘the’는 4번, ‘cat’은 2번, 그리고 나머지는 한번 나타나므로 발생빈도에 따라 ‘the’에 1, ‘cat’에 2, 나머지는 빈도가 동일하므로 표본의 차례대로 ‘sits’에 3, ‘on’에 4, ‘mat’에 5, ‘other’에 6, ‘runs’에 7, ‘over’에 8, ‘mats’에 9번을 부여한다.
- 빈도에 따라 인덱스를 부여하는 이유는 빈도가 높은 단어일수록 각 표본의 특성을 나타내는 데에 중요도가 낮아지거나 높아지기 때문이다. 그러므로 “The cat sits on the mat”는 [1,2,3,4,1,5]로 인덱스가 부여되고 “The other cat runs over the mats”는 [1,6,2,7,8,1,9]로 인덱스화 한다.
- 그러므로 “The cat sits on the mat”는 [1,2,3,4,1,5]로 인덱스가 부여되고 “The other cat runs over the mats”는 [1,6,2,7,8,1,9]로 인덱스화 한다.

Word2Vec과 Glove

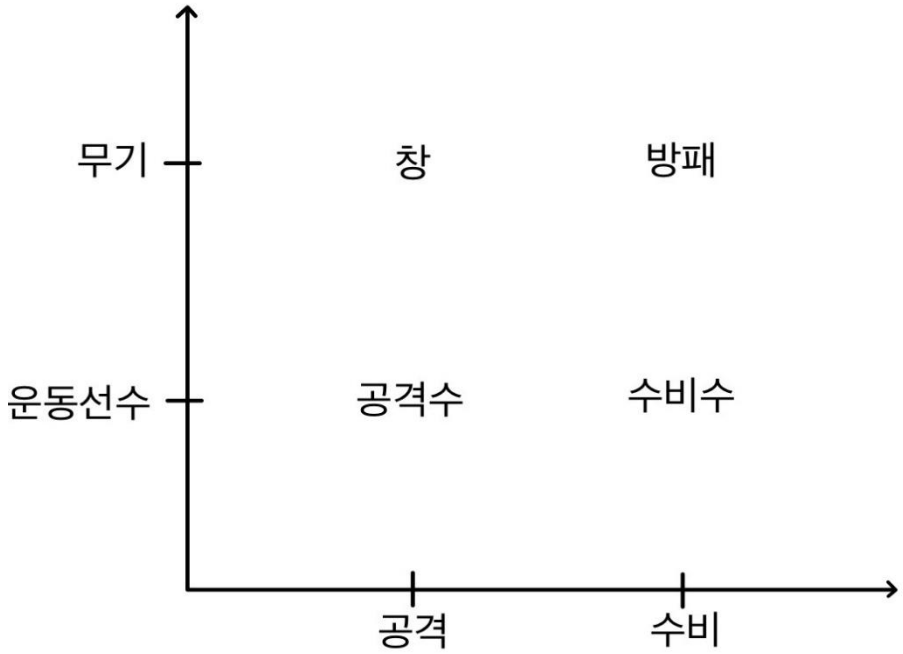
- 텍스트자료의 최종 수량화과정은 인덱스자료에 의미를 부여하는 과정이다.
- 신문자료와 같이 특정단어의 존재여부가 기사분류에 유용하면 다음과 같이 이항(binary) 인덱스로 전환한다.
- 즉, [1,2,3,4,1,5]를 [1,1,1,1,1,0,0,0,0]으로, [1,6,2,7,8,1,9]를 [1,1,0,0,0,1,1,1,1]로 자료를 전환한다.
- 즉, 첫 번째 열은 인덱스 1의 존재여부를, 2번째 열은 인덱스 2의 존재여부를,…, 그리고 9번째 열은 인덱스 9의 존재여부(존재하면 1, 그렇지 않으면 0)를 나타낸 것이다.
- 각 표본을 9개의 값으로 재 표현한 것으로 이 9개는 각 표본의 특성변수의 값으로 해석할 수 있다.

Word2Vec과 Glove

- 또 다른 수량화 과정은 자료의 빈도수로 표현할 수 있다.
- SNS에 있는 특정사안에 대한 댓글을 찬성과 반대로 분류하고자 할 때, 찬성과 반대에 빈번하게 나타나는 단어의 빈도가 중요한 특성변수의 값이 될 것이다.
- 첫 번째 표본에서는 인덱스 1이 2번, 인덱스 2, 3, 4, 5가 1번 나타났고 나머지 4개의 단어는 나타나지 않았으므로
- 첫 번째 표본은 [2,1,1,1,1,0,0,0,0]으로 수량화 되고 같은 방법으로 두 번째 표본은 [2,1,0,0,0,1,1,1,1]으로 수량화할 수 있다.
- 이러한 수량화도 표본을 동일한 크기의 9개의 특성변수로 재 표현한 것으로 볼 수 있다.

Word2Vec과 Glove

- 지금까지 논의한 텍스트의 수량화는 사람이 인지하는 단어들의 관계를 특성화하지 못하는 약점을 가지고 있다.



- 위 그림은 '창', '방패', '공격수', '수비수'라는 네 개의 단어를 '공격', '수비'라는 추상적인 개념과 '무기', '운동선수'라는 개념으로 재 분류하고 있다.

Word2Vec과 Glove

- 마치 사람과 같이 '창'과 '공격수'를 공격이라는 추상적인 개념으로 서로 연관되도록 수량화하고 '공격수'와 '수비수'를 운동선수라는 개념으로 연관시킬 수 있는 수량화가 필요하다.
- 또한 '여성'과 '왕'이라는 단어로 '여왕'이라는 단어를 추론되어야 하고, '아이'와 '아이들'은 단수와 복수의 관계로 이 두 단어가 거의 같다는 수량화가 필요하다.
- 이러한 수량화는 소위 word embedding을 통해 가능하며, word embedding은 인간의 언어를 단어 간의 연관성을 거리개념으로 전환하는 과정이라고 할 수 있다.

Word2Vec과 Glove

- Word embedding의 개념은 앞 그림의의 확장으로 생각하면 된다.
- 공격수는 '공격', '운동선수'라는 개념도 있지만 '축구', '프레미어리그' 등 여러 개념으로도 설명할 수 있듯이,
- 예를 들어, '공격수', '스트라이커'를 10개의 실수 값을 갖는 특성변수로 전환하여, 이 특성변수가 상관관계를 갖도록 하는 것이 word embedding의 목표이다.
- 단어들의 구문론적 의미를 부여하는 word embedding의 대표적인 embedding은 word2vec과 Glove (global vectors for word representation)이다.
- word2vec과 Glove는 딥러닝 모형이 아니며 비교적 간단한 은닉층 하나로 구성된 MLP 모형이다.

Word2Vec

- 이해를 돕기 위해 다음과 같은 간단한 두 문장(즉, 2개의 표본)만 가지고 word2vec의 개념을 설명하고자 한다.
- 'I am a good player', 'I am a nice player'
- 이 문장에서 'good'과 'nice'는 거의 동일한 의미를 가지고 있다. 우선 토큰화를 통해 단어를 분류한 후 유일한 단어에 인덱스를 부여하면
- {'i':1, 'am':2, 'a':3, 'good':4, 'player':5, 'nice':6}으로 정리할 수 있다.

Word2Vec

- 다음과 같이 입력자료(특성변수)와 출력자료(목적변수)를 정의해 보자.

입력	출력
am, a, good	i
i, a, good	am
i, am, good	a
i, am, a	good
am, a, good	player
am, a, nice	i
i, a, nice	am
i, am, nice	a
i, am, a	nice
am, a, nice	player

(a) CBOW 모형

입력	출력
i	am, a, good
am	i, a, good
a	i, am, good
good	i, am, a
player	am, a, good
i	am, a, nice
am	i, a, nice
a	i, am, nice
nice	i, am, a
player	am, a, nice

(b) Skip-gram 모형

Word2Vec

- word2vec embedding은 CBOW (continuous bag of words)와 skip-gram 두 개의 모형이 있다.
- CBOW는 출력단어에 인접한 3개의 단어(출력단어를 포함하여 4개, 이를 window 크기라고 한다)로 구성된 입력자료로 하나의 단어로 구성된 출력단어를 예측하는 입·출력층의 자료구조를 가지고 있고
- skip-gram 모형은 CBOW와 정반대의 입·출력층의 자료구조를 가지고 있다. 앞 그림에서 볼 수 있듯이 CBOW 모형에서 'good'과 'nice'는 동일한 입력자료를 가지고 있고 skip-gram 모형에서는 동일한 출력자료를 가지고 있으므로 'good'과 'nice'는 거의 동일한 구문론적 의미를 가지게 될 것이다(물론, 'nice'를 'bad'로 고치더라도 동일한 결과를 보일 것이다. 그러나 구문론적 의미는 수백만 개의 문장을 이용하므로 실제 문제에서는 이러한 오류는 발생하지 않는다).

Word2Vec

- 토큰화를 통해 생성된 유일한 단어들의 집합(이를 BOW (bag-of-words)라고 한다) 안의 단어들 각각을 one-hot 벡터로 전환한다.
- {'i':1, 'am':2, 'a':3, 'good':4, 'player':5, 'nice':6}이므로
- $i \rightarrow (1,0,0,0,0,0), am \rightarrow (0,1,0,0,0,0), a \rightarrow (0,0,1,0,0,0), good \rightarrow (0,0,0,1,0,0)$

  $player \rightarrow (0,0,0,0,1,0), nice \rightarrow (0,0,0,0,0,1)$로 벡터로 전환한다.
- 일반적으로 BOW의 크기가 V일 때(많게는 수백만개 이상), one-hot 벡터의 크기는 V이며 V개 단어 각각에 인덱스를 부여하고 이 인덱스 위치에만 1이고 나머지 V-1 인덱스들은 모두 0인 벡터이다.

Word2Vec

- BOW안에 있는 V개의 단어가 특성변수가 되며,
- CBOW는 1개 이상의 벡터화된 단어를 입력하고 하나의 벡터화된 단어를 예측하는 모형이고 skip-gram은 1개의 벡터화 단어를 입력하고 1개 이상의 벡터화 단어들을 예측하는 모형이라고 할 수 있다.
- CBOW와 skip-gram모형은 은닉층이 하나인 MLP모형이다.
- CBOW모형에서는 벡터화 입력자료의 평균을 입력하고 이를 은닉층에 전달하고 하나의 벡터화 단어를 출력하는 구조이다.
- skip-gram 모형에서는 하나의 벡터화 단어를 입력하고 이를 은닉층에 전달한 후 2개 이상의 벡터화 단어를 출력하는 구조이다.

Word2Vec

- 입력자료는 $V \times 1$이고 은닉층의 노드가 k개이면 입력자료는 $V \times k$ 모수로 선형결합하게 되며 이 모수를 W로 표기하고 embedding이라고 한다.
- 또한 CBOW와 skip-gram모형에서의 은닉층은 활성함수로 선형함수를 사용한다.
- 예제에서 CBOW는 'am','a','good'이 입력이고 'i'가 출력이므로

$$\frac{1}{3}[(0,1,0,0,0,0) + (0,0,1,0,0,0) + (0,0,0,1,0,0)] = (0,\frac{1}{3},\frac{1}{3},\frac{1}{3},0,0)$$가 입력되고

- 출력은 [1,0,0,0,0,0]이므로 총 6개의 단어 중 3개의 단어를 입력하여 첫 번째 단어를 예측할 확률을 최대화하는 W 를 구하는 문제가 된다.
- 그러므로 예를 들어 은닉층의 노드가 3개이면, W는 6×3 행렬이고 이를 입력변수에 곱하여 히든층은 (h_1, h_2, h_3)을 출력한다.
- (h_1, h_2, h_3)은 출력층에 입력되고 softmax함수에 의해 6개 단어에 속할 확률을 구한다.

Word2Vec

- skip-gram 모형도 CBOW 모형과 동일한 구조를 가지고 있기 때문에,
- 동일한 방법으로 크기가 $V \times k$인 embedding W를 계산하고, 은닉층과 출력층을 연결하는 모수도 동일한 $k \times V$행렬이다. k개의 노드가 출력층에 입력되고 softmax함수를 적용하여 다음의 확률을 계산한다.

$$P(I_1, I_2, \cdots, I_s | I_0) = \Pi_{k=1}^{s} \frac{\exp(u_k)}{\sum_{j=1}^{V} u_j}$$

- CBOW모형은

$$P(I_0 | I_1, I_2, \cdots, I_s) = \frac{\exp(u_c)}{\sum_{j=1}^{V} u_j}$$

- softmax함수를 사용하여 목적변수에 대응하는 단어에 속할 확률을 추정하지만 skip-gram 모형은 손실함수가 categorical cross entropy의 합(s개의 목적단어의 합)이 된다.

GLOVE (global vectors for word representation)

- GLOVE는 word2vec과 다르게 단어들 간의 빈도수를 이용한 word embedding이다.
- 간단한 예를 통해 기본적인 구조를 이해하도록 하자.
- 두 개의 문장(그러므로 2개의 표본) 'you are a girl', 'you are a boy'가 있을 때 토큰화를 통해 bag-of-words를 만들면 {'you', 'are', 'a', 'girl', 'boy'}가 된다.
- 이를 이용하여 두 문장에서 발생한 발생빈도를 정리하면

	you	are	a	girl	boy
you	0	2	2	1	1
are	0	0	2	1	1
a	0	0	0	1	1
girl	0	0	0	0	0
boy	0	0	0	0	0

GLOVE (global vectors for word representation)

- ‘you’와 ‘are’는 두 문장에서 모두 발생하므로 2, ‘you’와 ‘a’도 두 문장 모두에서 발생하므로 2, 그리고 ‘you’와 ‘girl’은 첫 문장에서만 발생하므로 빈도수 1을 갖는다.
- 두 문장에서 다 나타나지 않으면 0의 빈도를 가진다.
- 그러나 위 빈도는 해당 단어의 오른 쪽에 나타난 빈도수이므로 거리의 개념이 없다. 그리하여 단어들 간의 떨어진 단어 간격으로 나누어 단어의 거리를 조절할 필요가 있다.

	you	are	a	girl	boy
you	0	2	1	0.33	0.33
are	0	0	2	0.5	0.5
a	0	0	0	1	1
girl	0	0	0	0	0
boy	0	0	0	0	0

GLOVE (global vectors for word representation)

- 여기에 추가적으로 window를 적용하여 단어간격이 2 이상이면 가중치를 0으로 재조정하면

	you	are	a	girl	boy
you	0	2	1	0	0
are	0	0	2	0.5	0.5
a	0	0	0	1	1
girl	0	0	0	0	0
boy	0	0	0	0	0

- 실제 문제에서는 총 단어수가 V개 이므로, 단어의 동시 발생빈도표는 $V \times V$ 가 된다.
- 단어 수가 많아지면 동일단어간의 단어거리가 표본마다 다를 수 있다. 이 경우, 표본마다 window를 적용하여 window 크기보다 단어거리가 긴 것을 0으로 먼저 조정한 후, 이 조정된 빈도를 합산하여 발생빈도를 구하면 된다.

GLOVE (global vectors for word representation)

- GLOVE의 손실함수는

$$\ell = \sum_{i,j=1}^{V} f(x_{ij})(\boldsymbol{w}_i^T \boldsymbol{w}_j + b_i + b_j - \log x_{ij})$$

여기에서 x_{ij}는 i번째 단어와 j번째 단어의 동시 발생빈도이며

$$f(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0 \\ \frac{x_{ij}}{100} & \text{if } x_{ij} < 100 \\ 1 & \text{if } x_{ij} \geq 100 \end{cases}$$

- $\begin{pmatrix} \boldsymbol{w}_i \\ b_i \end{pmatrix}$는 i번째 단어의 word embedding이다.
- word2vec과 GLOVE의 실제 아키텍쳐는 논의하지 않고자 한다.
- 그 이유는 word2vec과 GLOVE는 이전학습(transfer learning)을 위한 사전학습모형(pre-training model)으로 주로 사용되기 때문이다.
- word2vec은 3,000,000개의 단어로 학습된 word embedding이고 GLOVE는 400,000개의 단어로 학습된 word embedding을 제공하고 있다.

텍스트자료에 특화된 Word Embedding

- word2vec이나 GLOVE에 의한 word embedding은 단어들 간의 구문론적 의미를 비교적 잘 전달하지만 인간의 언어는 환경과 문화에 따라 그 의미가 다른 경우가 많다.
- 예를 들어, 법률에 관련된 텍스트와 연애소설에 관련된 텍스트는 동일한 언어를 쓰더라도 단어들의 구조나 구성이 매우 다르고 의미 또한 다를 경우가 많다.
- 이러한 배경에 의해 분석대상 텍스트에 특화된 word embedding이 필요한 경우가 많다.
- 텍스트에 특화된 word embedding은 근본적으로 word embedding과 동일한 구조를 가지고 있다.

텍스트자료에 특화된 Word Embedding

- 앞에서 논의한 word2vec에서는 전체 표본에 있는 유일한 단어의 수만큼의 크기를 가진 one-hot coding을 이용하여 각 단어를 수량화한 바 있다.
- 그러므로 전체 표본에서 차지하는 단어수가 많으면 단어를 나타내는 one-hot coding의 크기도 클 수밖에 없다.
- 즉, 각 단어는 자신의 index에 해당하는 위치만 1값을 가지고 있고 나머지는 모두 0으로 구성되어 있는 희소벡터(sparse vector)로 표현된다.
- 분석 텍스트에 특화된 word embedding은 각 단어를 밀집벡터(dense vector)로 표현하는 기법이다.
- 예를 들어, 총 단어가 5개있을 때 첫 단어를 [1,0,0,0,0]로 재 표현한 것이 희소벡터이고 [0.2, 0.7]로 표현한 것이 밀집벡터이다. 밀집벡터를 이용하면 희소벡터보다 훨씬 적은 크기의 벡터로 각 단어를 표현할 수 있으며,

텍스트자료에 특화된 Word Embedding

- 텍스트자료 분석의 목적에 따라 동일한 단어도 다른 밀집벡터로 표현할 수 있는 유연성을 가지게 된다.
- 텍스트분석을 위해서 각 표본은 동일한 텐서로 구성되어야 한다. 텍스트자료의 표본(문서 또는 문장)에 있는 단어의 수가 같지 않으므로 각 표본의 길이를 동일하게 하기 위해 padding을 이용한다.
- padding은 표본내의 단어 길이가 정해진 길이보다 크면 잘라내고, 모자라면 0으로 채워넣는 과정을 말한다. padding과정에서 0은 결측치로 인식하게 된다.
- padding에 의해 모든 표본의 길이가 m개로 통일되었고 각 단어를 크기가 k인 밀집벡터로 word embedding하면 각 표본은 크기가 (m,k) 인 2D텐서자료가 된다.
- 즉, m은 RNN모형에서 정의된 시간스탭(timestep)이 되고 k는 특성변수수가 된다.

Word Embedding 사례분석

- 지금까지 논의한 텍스트자료의 토큰화, 단어에 인덱스 부여하기, 각 표본의 단어를 인덱스의 계열(sequence)표현하기, 그리고 padding 등의 과정을 아래와 같이 간략한 예로 구현해 보기로 하자.

Word Embedding 사례분석

- IMDB (internet movie database)는 인터넷에 50,000개의 영화평으로 영화평가에 "긍정적", "부정적"으로 분류된 텍스트자료이다.
- 자료분석의 기술적 측면에서, 자료분석과정은 자료의 사전정리, 모형설정, 모형검증, 모형적용의 순서를 따른다.
- 그러나 자료분석의 개념적 측면에서는 정반대의 과정을 따라야 한다. 즉, 모형적용을 위해서 연구목적과 기대효과를 설정하고 이에 대한 해결책으로 머신러닝 모형을 쓸 것인지 딥러닝 모형을 쓸 것이지를 결정해야 한다.
- 기대하는 응용결과에 부합하기 위한 기대하는 모형의 성능을 미리 설정해 놓아야 한다.
- 분석도구(즉, 모형)가 결정되면 이 모형에 사용가능한 자료를 구한 후, 자료의 사전정리과정을 시행하여야 한다.

Word Embedding 사례분석

- 그러므로 data scientist가 되기 위해서는 머신러닝이나 딥러닝모형의 이해와 모형적합 뿐만 아니라, 모형에 적합한 자료를 만들어 내는 능력도 필수적이라고 할 수 있다.
- 이를 위해 이미 사용해본 IMDB 원 데이터를 직접 내려받아 텍스트자료의 실수화를 직접 실습해보고 난 후,
- GLOVE를 사전학습모형(pre-trained model)로 하는 이전학습을 논의하고자 한다.

Q & A