単語ベクトル

正田 備也

masada@rikkyo.ac.jp

cf. Speech and Language Processing: New Tools, New Applications.

Tutorial by DiDi Chuxing @ ICDM 2019

https://outreach.didichuxing.com/internationalconference/icdm2019/tutorial/nlp-speech.pdf

NLP problems

- Sentiment classification
- Machine translation
- Question answering
- Pronoun resolution
- Spelling correction
- Entity tagging / linking
- Parsing
- Relation extraction
- Summarization
- Word segmentation

- これらの問題を教師あり学習で解く
- 何を教師信号にする？

1. 文や文書にタグづけする
   - 文書のカテゴリ（政治、経済、スポーツ、…）
   - 文書が表す感情（ポジ／ネガ）
2. 単語列を別の列に変換
   - どのような列に変換すべきかを教師信号として用意する

Entity tagging (Named entity recognition)

- named entity
  - 固有名詞で指示されるもの（人、場所、企業、組織、etc）

John went to New Orleans

PER --- --- LOC LOC

Machine translation

I like to eat apples.

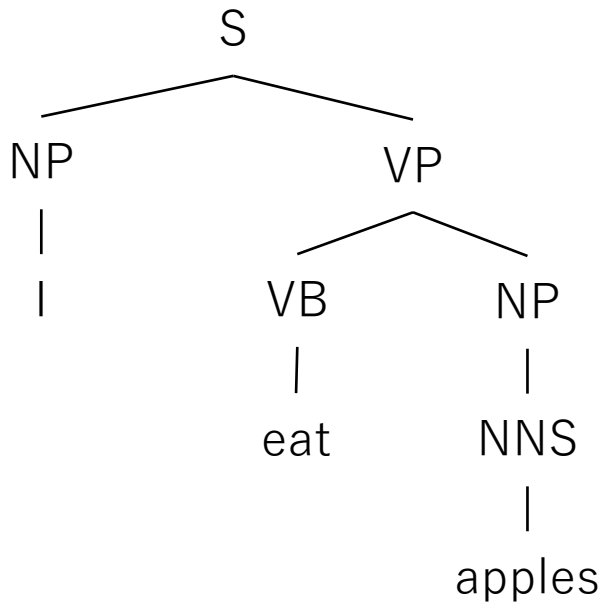
我喜欢吃苹果。

Parsing

https://web.stanford.edu/~jurafsky/slp3/11.pdf

I eat apples.

(S (NP I) (VP (VB eat) (NP (NNS apples))))



S = sentence
NP = noun phrase
VP = verb phrase
VB = verb (base form)
NNS = plural noun

Sequence-to-sequence models

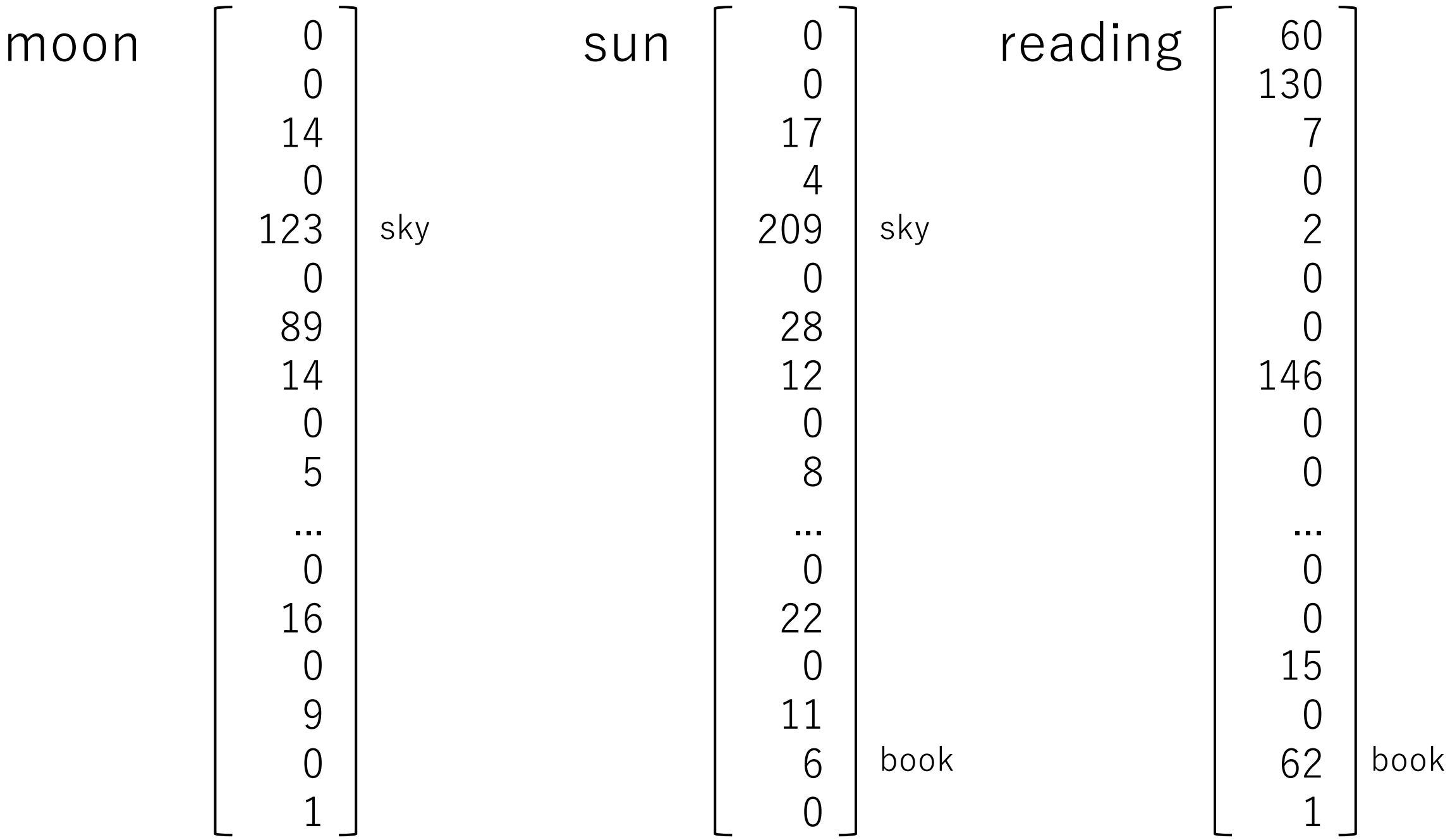
- 現代的な自然言語処理はseq2seqモデルで解かれることが多い
- seq2seqモデル = ある単語列を別の単語列へ変換するモデル
- これは非常に一般的な枠組み
  - 様々な問題をseq2seq型の問題として再定式化できる

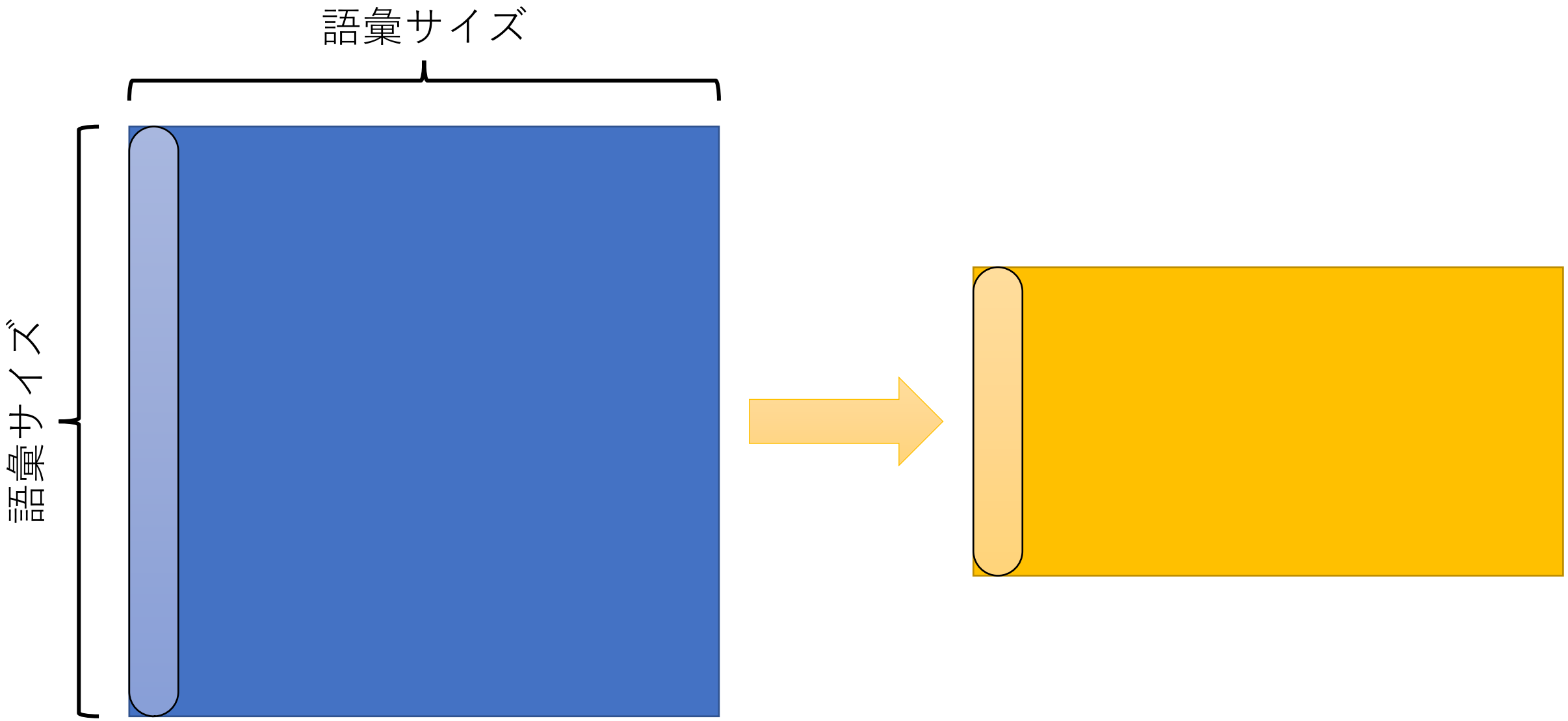
- まず、アイテムの列をベクトルの列として表現する
  - あるベクトル列を別のベクトル列へ変換する問題として定式化
  - つまり、個々のアイテムをベクトルとして表現する（embedding）

自然言語のデータ＝単語(?)の列

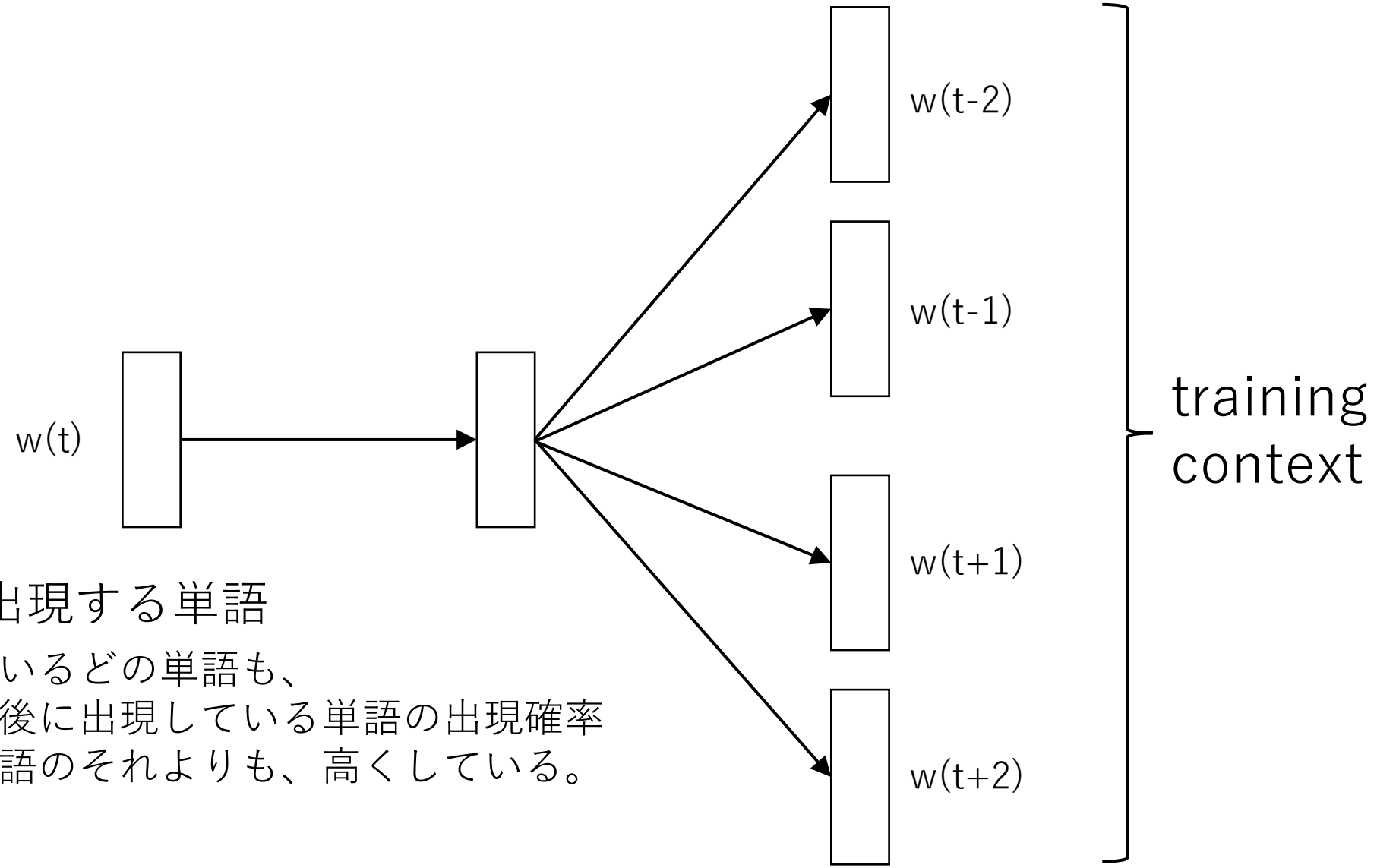
- 単語をどうやってembedするか？
  - https://www.aclweb.org/anthology/P14-1023.pdf

1. コーパス内での他の単語との共起頻度で表現
   - 例：前後１０単語を見て共起した回数を数える
   - 問題点：ベクトルの次元が語彙サイズになる
     - 次元圧縮する
2. Word2Vec [Mikolov+ 13]
   - https://arxiv.org/abs/1301.3781
   - https://arxiv.org/abs/1310.4546


moon
0
0
14
0
123
sky
0
89
14
0
5
...
0
16
0
9
0
1
sun
0
0
17
4
209
sky
0
28
12
0
8
...
0
22
0
11
6
book
0
reading
60
130
7
0
2
0
0
146
0
0
...
0
0
15
0
62
book
1


語彙サイズ
語彙サイズ

word2vec – Skip-gram model [arXiv:1310.4546]



w(t)：t番目に出現する単語

テキストに現れているどの単語も、
そのテキストで前後に出現している単語の出現確率
を、それ以外の単語のそれよりも、高くしている。

Skip-gram modelで最大化する関数

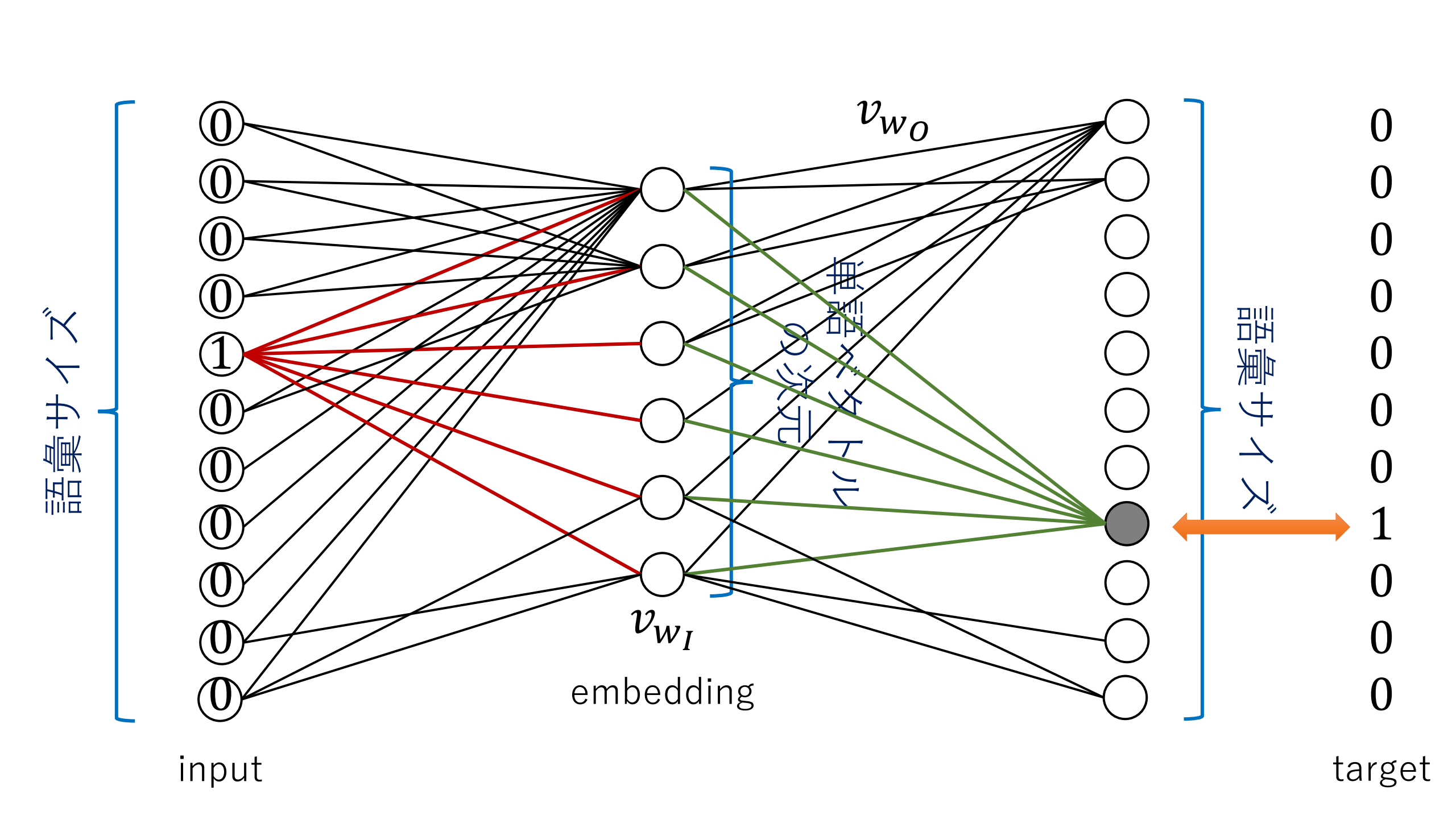
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

- 前後c個の単語をコンテクストとみなす
- 条件付き確率$p(w_{t+j}|w_t)$は以下のように計算する

$$p(w_O|w_I)=\frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})}$$

規格化項を求めるのが大変！

$$\therefore \log p(w_O|w_I)={v'_{w_O}}^T v_{w_I}-\log\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})$$


語彙サイズ
0
0
0
0
1
0
0
0
0
0
0
v_{w_O}
v_{w_I}
embedding
隠れ層ニューロンの次元
語彙サイズ
0
0
0
0
0
0
1
0
0
0
input
target

Skip-gramに加えられる変更

- 最大化する関数を変更（規格化項を求めずにすむように＋α）

$$\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} \mathrm{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_w}^T v_{w_I}) \right]$$

- $P_n(w)$は単語wの出現頻度の3/4乗に比例
- Subsampling
  - 頻出語のトークンは訓練データから外す
    - 一定の確率で、外す

Subword embedding

https://arxiv.org/pdf/1609.08144.pdf

- OoV(out-of-vocabulary)問題
  - 上のアプローチだと、未知語に対処できない
  - どんな単語が出てきてもそれに対応するembeddingが得られるようにするには、単語より細かい単位をembedすればよい
    - 語彙サイズは固定しつつ、コーパスをできるだけ効率よく（＝できるだけ少ない数のトークンへ）分割するサブワード群を求めたい。

- **Word**: Jet makers feud over seat width with big orders at stake
- **wordpieces**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

In the above example, the word “Jet” is broken into two wordpieces “_J” and “et”, and the word “feud” is broken into two wordpieces “_fe” and “ud”. The other words remain as single wordpieces. “_” is a special character added to mark the beginning of a word.

word vectorsの現在

word vectorsから

contextualized word vectorsへ

・・・いずれまた説明します。