NLP2팀 2주차(2020.04.23.)

한국어 임베딩
4. 단어 수준 임베딩
4.7~4.8 (151p~171p)

황인택

차례

4.7 어떤 단어 임베딩을 사용할 것인가

4.7.1 단어 임베딩 다운로드

Word2Vec / FastText / Glove / Swivel

한국어 위키백과 / 네이버 영화 리뷰 말뭉치 / KorQuAD 말뭉치

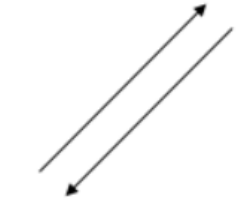
차원 수 = 100

4.7 어떤 단어 임베딩을 사용할 것인가
4.7.2 단어 유사도 평가

일련의 단어 쌍을 미리 구성한 후에,
사람이 평가한 점수와 단어 벡터 간 코사인 유사도 사이의 상관관계를 계산함으로써
단어 임베딩의 품질을 평가하는 방법

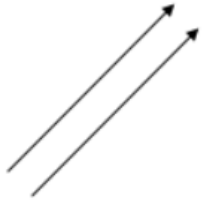
- 코사인 유사도

$$similarity = cos(\Theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$



코사인 유사도 : -1

코사인 유사도 : 0



코사인 유사도 : 1

준비물:
사람이 평가한 평가 데이터셋 >> 단어 유사도 평가 데이터셋(이동준 외, 2018)
학습이 완료된 단어 임베딩 >> class WordEmbeddingEvaluator

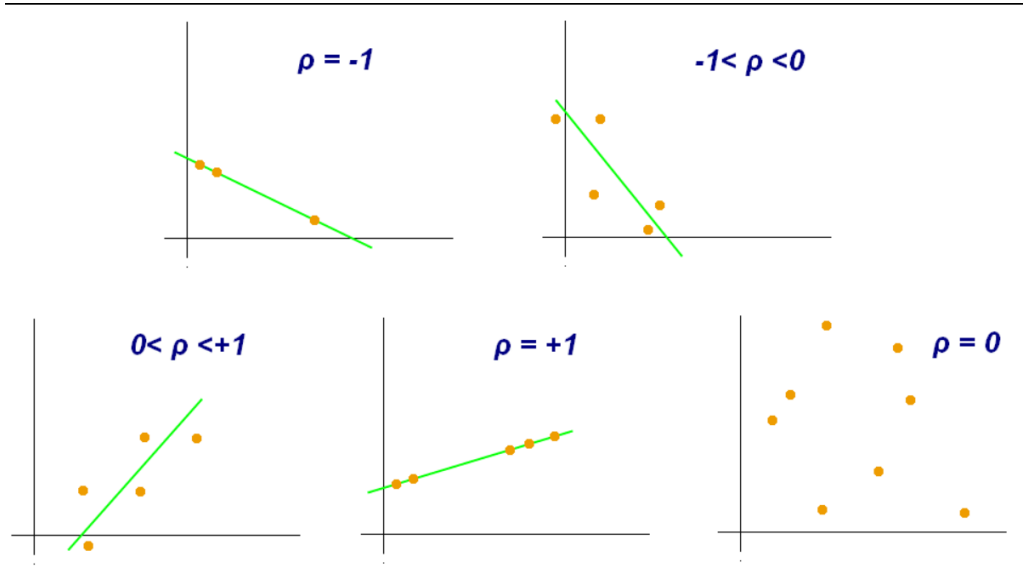
평가 지표

피어슨 상관계수: 두 변수 X와 Y간의 선형 상관 관계를 계량화한 수치

$$r_{XY} = \frac{\frac{\sum_i^n (X_i - \overline{X})(Y_i - \overline{Y})}{n-1}}{\sqrt{\frac{\sum_i^n (X_i - \overline{X})^2}{n-1}} \sqrt{\frac{\sum_i^n (Y_i - \overline{Y})^2}{n-1}}}$$

두 변수의 표준편차의 곱

두 변수의 공분산



스피어만 (순위)상관계수: 순위가 매겨진 변수 간의 피어슨 상관 계수

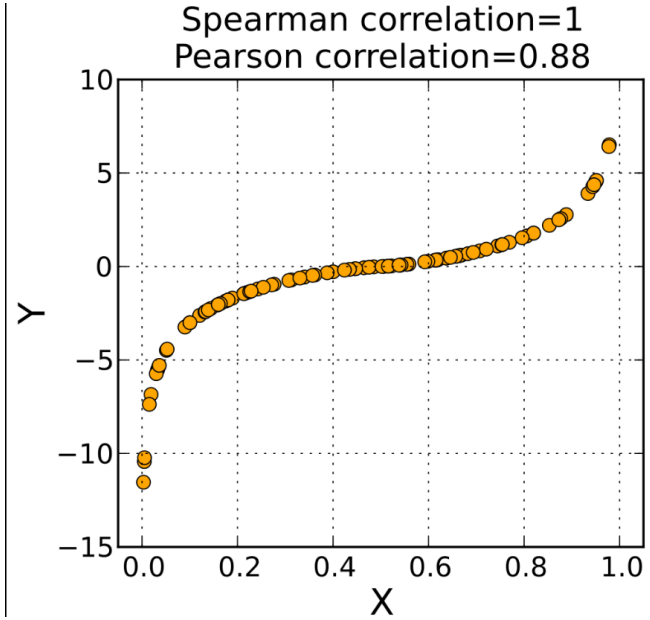
$$\hat{\rho} = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2} \cdot \sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}}$$

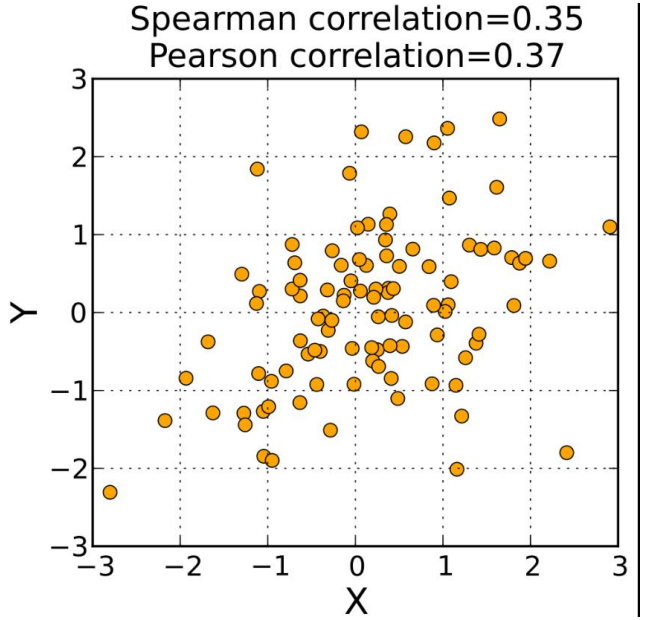
→

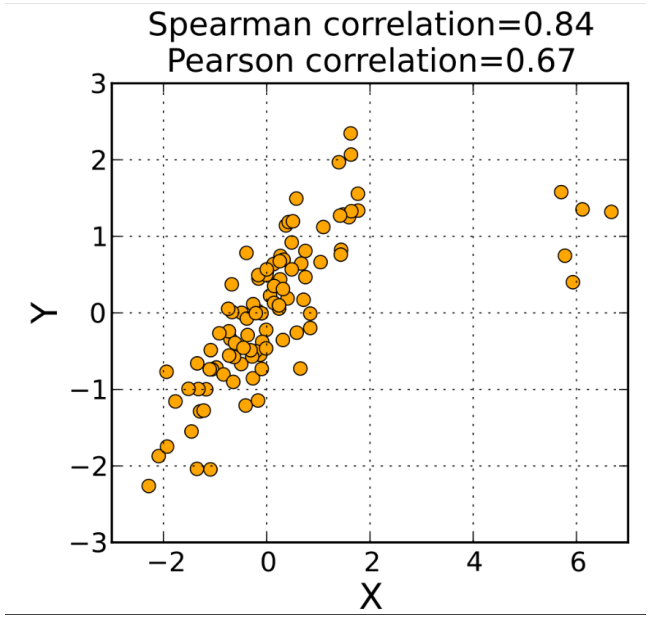
$$r_s = \frac{\sum_{i=1}^{n} (R_i - \overline{R})(S_i - \overline{S})}{\sqrt{\sum_{i=1}^{n} (R_i - \overline{R})^2} \cdot \sqrt{\sum_{i=1}^{n} (S_i - \overline{S})^2}}$$

R,S는 X,Y를 오름차순으로 정리한 것

피어슨과 스피어만의 관계





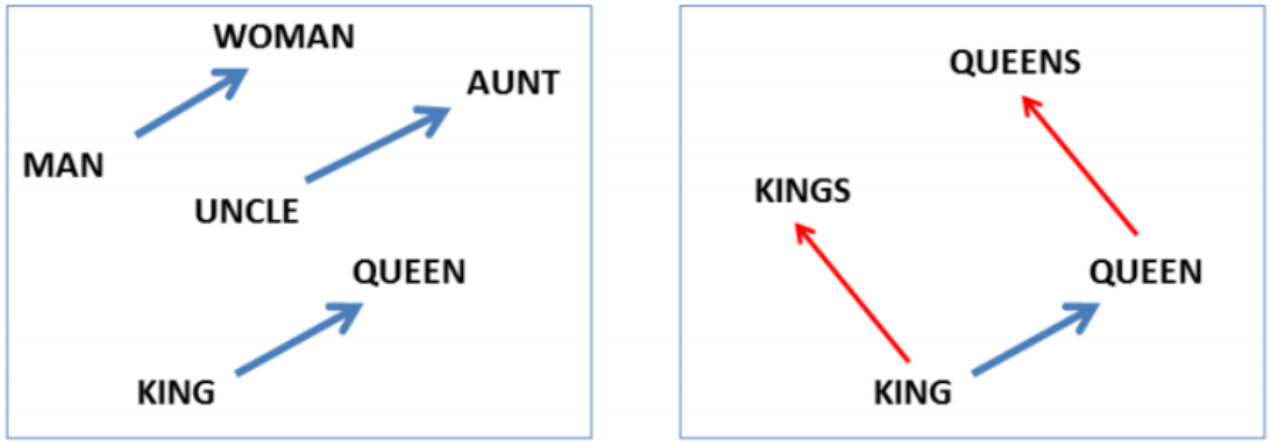


```
>>> model_name='word2vec'
>>> model.word_sim_test('data/raw/kor_ws353.csv')
spearman corr: 0.5770993871014621 , pearson corr: 0.5956751142850295
```

```
>>> model=WordEmbeddingEvaluator(vecs_txt_fname='data/word-embeddings/swivel/row_embedding.tsv',method='swivel',dim=100,
tokenizer_name='mecab')
>>> model.word_sim_test('data/raw/kor_ws353.csv')
spearman corr: 0.549541215508716 , pearson corr: 0.5727286333920304 , # of errors: 0
```

4.7 어떤 단어 임베딩을 사용할 것인가
4.7.3 단어 유추 평가

의미론적으로 단어 벡터 간 계산을 통해 유추하는 방법



(Mikolov et al., NAACL HLT, 2013)

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Example Word Pairs for Analogy Reasoning Task

준비물:
단어 유추 평가 데이터셋 (이동준 외, 2018) – Google analogy 참고함.
학습이 완료된 단어 임베딩 >> class WordEmbeddingEvaluator

Word2vec 모델로 평가한 예시

```
>>> from models.word_eval import WordEmbeddingEvaluator
>>> model=WordEmbeddingEvaluator(vecs_txt_fname='data/word-embeddings/word2vec/word2vec',method='word2vec',dim=100,token
izer_name='mecab')
/usr/local/lib/python3.5/dist-packages/smart_open/smart_open_lib.py:398: UserWarning: This function is deprecated, use s
mart_open.open instead. See the migration notes for details: https://github.com/RaRe-Technologies/smart_open/blob/master
/README.rst#migrating-to-the-new-open-function
  'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
>>> model.word_analogy_test('data/raw/kor_analogy_semantic.txt',verbose=True)
대한민국 - 서울 + 일본
correct answer: 도쿄
predicted answers: [('대한민국', 0.6562236), ('미국', 0.6553047), ('한국', 0.64585644), ('중국', 0.5824025), ('나라', 0.
57273895), ('영국', 0.5665904), ('민국', 0.53613776), ('인민공화국', 0.528635), ('태국', 0.51655376), ('필리핀', 0.51417
67), ('무조건', 0.50976455), ('입맛', 0.5086572), ('베트남', 0.5042817), ('민주주의', 0.50088084), ('타이완', 0.49794406
), ('인도', 0.4977575), ('방공식별구역', 0.49372968), ('이래서', 0.49224794), ('외국', 0.48906964), ('각국', 0.4870873),
 ('대만', 0.48654234), ('인도네시아', 0.48586324), ('러시아', 0.48545858), ('제국주의', 0.48544908), ('뉴질랜드', 0.4839
6832), ('열도', 0.4838866), ('수출국', 0.4832701), ('오키나와현', 0.47963917), ('중화', 0.47755918), ('국방백서', 0.4771
0842)]
```

```
>>> model.word_analogy_test('data/raw/kor_analogy_semantic.txt',verbose=False)
# of correct answer: 158 , # of data: 420 , # of errors: 0
```

4.7 어떤 단어 임베딩을 사용할 것인가
4.7.4 단어 임베딩 시각화 – word2vec 기준, t-SNE로 시각화

t-Stochastic Nearest Neighbor (t-SNE)

고차원의 벡터로 표현되는 데이터 간의 neighbor structure를 보존하는
2 차원의 embedding vector 를 학습함으로써,
고차원의 데이터를 2 차원의 지도로 표현

How to embed high dimensional vectors to low dimensional space

고차원의 원 공간에서 저차원의 임베딩 공간으로 데이터의 공간을 변환하기 위해서 t-SNE 는 원 공간에서의 데이터 간 유사도 p_{ij} 와 임베딩 공간에서의 데이터 간 유사도 q_{ij} 를 정의합니다.

$$p_{j|i} = \frac{exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-|x_i - x_k|^2/2\sigma_i^2)}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

$$q_{ij} = \frac{(1 + |y_i - y_j|^2)^{-1}}{\sum_{k \neq l} (1 + |y_k - y_l|^2)^{-1}}$$

역수의 무한대를 막기 위해서 1을 더함.

점마다 시그마 값이 다르기 때문에 두 확률 값의 평균으로 유사도를 정의하되,
1/n을 곱하여 모든 점 간의 유사도 합이 1이 되도록 scaling함.

>> 시그마의 기준은 Perplexity 파라미터이며,
이는 학습의 영향을 주는 점들의 개수

Gradient descent를 이용해 p에 가깝도록 q를 학습함. 즉, q를 정의하는 y1,y2를 학습함.(q는 y1,y2의 유사도)

$$\frac{\delta C}{\delta y_i} = \sum_j (p_{ij} - q_{ij})(y_i - y_j)\frac{1}{1 + |y_i - y_j|^2}$$

P와 q의 차이가 적을 수록 이동하지 않음.

4.8 가중 임베딩

4.8.1 모델 개요

문서 내 단어의 등장은 저자가 생각한 주제에 의존한다고 가정 >> 주제 벡터(discourse vector)

주제 벡터 c가 나타났을 때 단어 w가 나타날 확률

$$P(w|c_s) = \alpha P(w) + (1-\alpha) frac exp(\tilde{c_s} \cdot v_w) Z$$

주제 벡터 c가 나타났을 때 단어의 시퀀스인 문장 s가 나타날 확률

$$P(s|c_s) \propto \sum_{w \in s} log P(w|c_s) = \sum_{w \in s} f_w(\tilde{c_s})$$

문장 s 안의 단어 등장 확률의 테일러 근사

$$f_w(\tilde{c_s}) \approx f_w(0) + \nabla f_w(0)^T \tilde{c_s}$$

$$= constant + frac(1-\alpha)/\alpha Z P(w) + (1-\alpha)/\alpha Z \tilde{c_s} \cdot v_w$$

문장이 등장할 확률을 최대화하는 주제벡터는, 문장에 속한 단어들에 해당하는 단어벡터에 가중치를 곱해 만든 새로운 벡터들의 합에 비례함.