문장벡터를 이용한 동형어, 다의어 구분과 한시 분류

박진호(서울대 국문과)

목차

문제의식

- 텍스트를 읽을 때, 이 텍스트가 다른 텍스트와 비슷한 점이 있다는 느낌을 받을 때가 있음.
- 이러한 유사성에 대한 감각은 표면적인 유사성(예컨대 사용된 단어가 같거나 비슷함)에 기인할 수도 있고, 보다 심층적인 유사성(예컨대 비슷한 의미를 나타냄)에 기인할 수도 있음.
- 이러한 유사성을 단순히 주관적인 감각에만 의존해서 판단하기보다는 더 객관적인 지표를 사용하는 게 바람직.
- 객관적, 계량적인 유사성 지표는 표면적 특성을 바탕으로 해서 비교적 손쉽게 고안할 수 있으나, 이런 지표는 보다 심층적, 의미적인 특성을 포착하지 못한다는 단점이 있음.
- 최근 활발하게 연구되고 있는 인공신경망에 바탕을 둔 word2vec 같은 알고리즘을 이용하면 보다 심층적, 의미적인 유사성을 포착할 수 있음.
- 이러한 방법을 이용하여 동형어, 다의어 의항 구분, 한시 분류 등을 시도해 보고자 함.

1.1. 표면적 유사도 지표: 편집거리

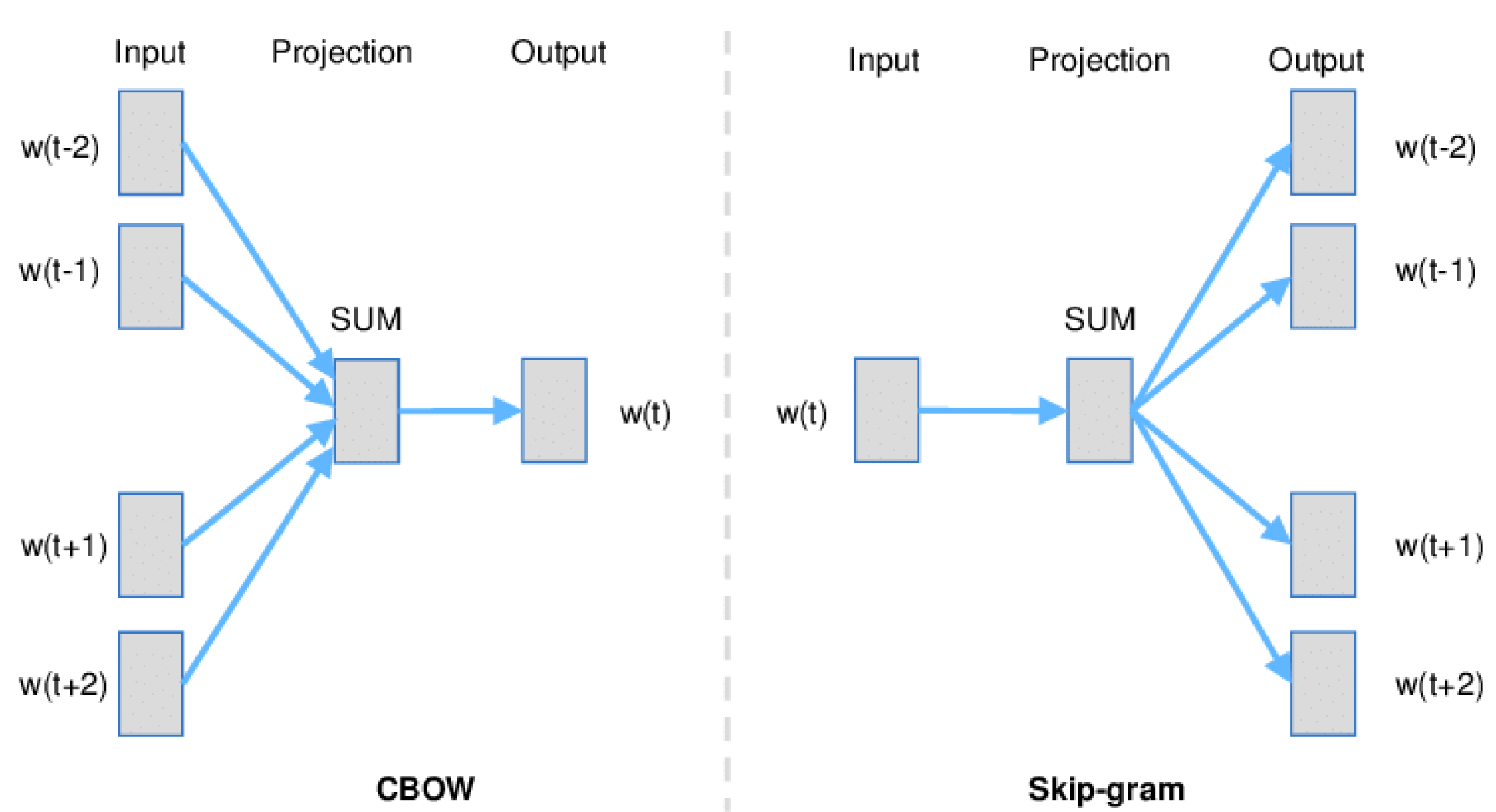
- X와 Y라는 2개의 문자열이 있을 때, X를 Y와 동일한 문자열로 만들기 위해 소요되는 조작(operation)의 개수를 바탕으로 하여 편집거리를 계산.
- 조작의 종류: 삭제(deletion), 삽입(insertion), 대치(substitution)
- 삭제와 삽입에 비해 대치는 더 복잡한 조작이므로(삭제+삽입이라고 볼 수 있음), 삭제와 삽입에는 1, 대치에는 2의 점수를 부여.
- 동일한 두 문자열 사이의 편집거리는 0, 유사도는 1
- “abcdef”와 “ghijh”라는 두 문자열은 공통 부분이 전혀 없으므로 유사도는 0, 편집거리는 2*6=12
- “abcdef”와 “abdxyf”라는 2개의 문자열이 있을 때, 후자의 “ab” 뒤에 “c”를 삽입하고(1점) “x”를 “e”로 대치하고(2점) “y”를 삭제하면(1점) 전자와 동일하게 됨. 이 두 문자열 사이의 편집거리는 4
- “abcdef”와 “abdxyf”는 길이가 6이므로 두 문자열 사이의 유사도는 (1 – 4/(6+6)) ≒ 0.67 (6개의 글자 중 4개가 같으므로 직관과 부합)

편집거리의 한계와 대안

- 편집거리는 의미적 유사성을 반영하지 못한다는 단점이 있음.
- “思故乡”과 “想旧家”는 공통된 글자가 하나도 없으므로 편집거리 6, 유사도 0 (= 1 – 6/(3+3))으로 판정되지만, 사실은 매우 비슷한 의미를 나타냄.
- 문자열의 표면적 비교에 머무르지 않고, 심층적인 의미의 유사성을 측정할 수 있는 방법이 필요함.

1.2. 언어요소의 벡터화 방법

- 최근 인공지능 기술의 핵심인 기계학습(특히 딥러닝)이 이미지, 음성, 텍스트 처리에서 각광을 받고 있음.
- 기계학습으로 언어를 처리하려면 언어 단위(글자, 단어, 문장 등)를 수치 벡터로 나타내야 함.
- 2012년경 효율적으로 언어 단위를 벡터화하는 알고리즘이 Mikolov 등에 의해 개발됨: Word2Vec
- 주위 문맥을 반영하도록 언어요소(단어)를 벡터화.
- window/span의 크기를 일정하게 정해 놓고(예컨대 5), 타깃 단어의 앞뒤 window 내에 출현하는 단어들이 비슷하면 비슷할수록 비슷한 벡터로 표상되게끔 함.


Input
Projection
Output
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
Input
Projection
Output
w(t)
SUM
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Word2Vec의 2가지 알고리즘 (1): Continuous Bag of Words (CBOW)

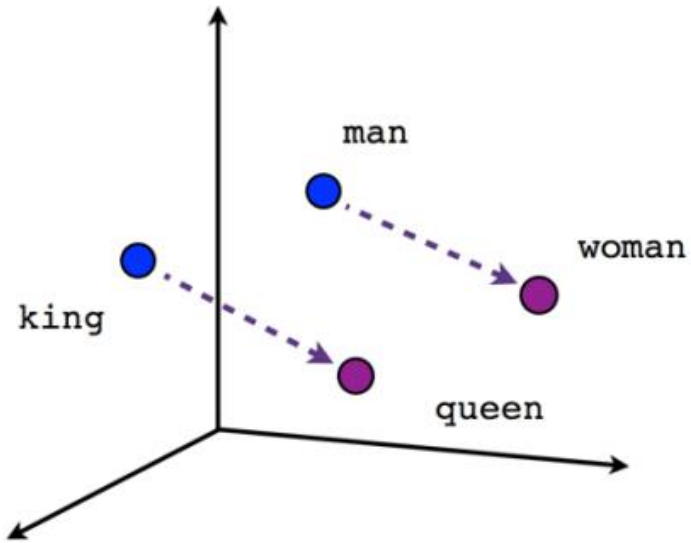
- 앞뒤 window 내에 출현하는 단어들을 모델에게 알려주고 타깃 단어가 무엇인지 알아내게끔 모델을 훈련시킴.
- 대규모 말뭉치를 반복적으로 관찰하면서 모델은 앞뒤 문맥으로부터 타깃 단어를 더 정확하게 알아내려고 노력.
- 이를 위해 각 단어의 벡터 수치를 조금씩 조정.
- 이 과정을 반복할수록 모델의 예측의 정확도가 올라감.
- 우리의 관심은 이 예측 자체가 아니고, 이 훈련의 부산물로 얻어진 각 단어의 벡터 표상

Word2Vec의 2가지 알고리즘 (2): Skip-gram

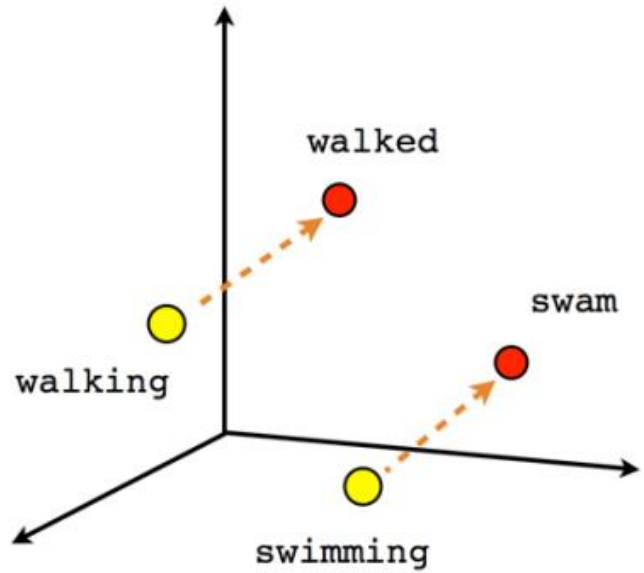
- CBOW의 경우와는 반대로, 타깃 단어를 모델에게 알려주고, 그 앞뒤의 window 내에 출현하는 단어들을 알아맞히게끔 모델을 훈련시킴.
- 앞의 5 단어, 뒤의 5 단어를 주고 그 사이에 나오는 타깃 단어를 알아맞히게끔 하는 CBOW에 비해
- skip-gram은 타깃 단어 하나만 주고 그 앞뒤의 10개 단어를 알아맞히라는 것이니 난이도가 훨씬 높음.
- 우리의 관심은 이 예측의 정확도 자체가 아니라, 그 부산물로 얻어지는 단어의 벡터 표상.
- CBOW보다 Skip-gram 알고리즘에 의해 얻어진 벡터 표상이 단어 의미를 더 잘 나타낸다고 알려져 있음.
  - 더 어려운 과제를 주어 빡세게 훈련시켰기 때문에, 일을 더 잘 하는 듯.

Word2Vec의 부산물: 단어 벡터

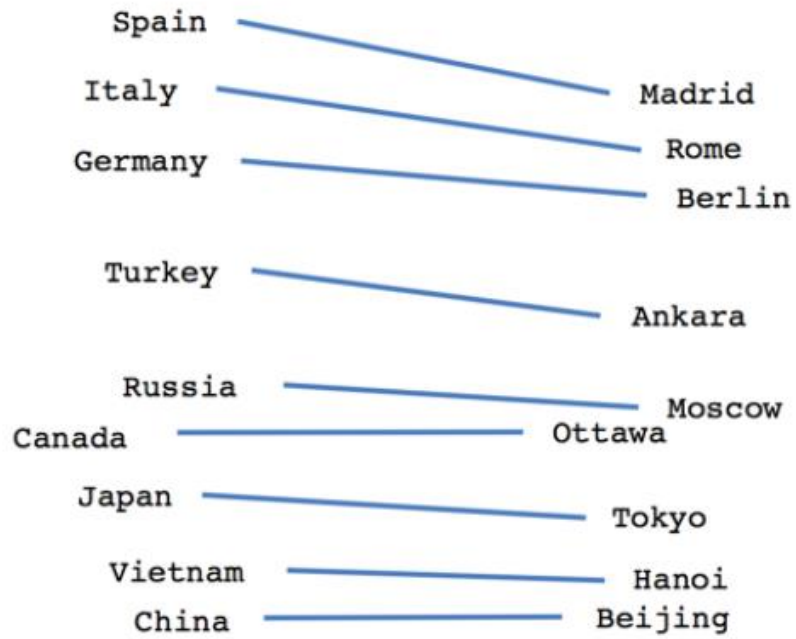
- Word2Vec 알고리즘에 대량의 언어 자료를 집어넣어 신경망을 훈련시키면, 그 결과로서 얻어지는 단어 벡터는 정말 의미상 유사한 단어들을 유사한 벡터로 표상함.
- 또한 A와 B 두 단어 사이의 의미 관계와 C와 D 두 단어 사이의 의미 관계가 평행(parallel)하면, A와 B 두 벡터 사이의 관계와 C와 D 두 벡터 사이의 수학적 관계도 평행함.
- 이 덕분에 벡터 연산으로 의미 연산을 모델링할 수 있음.
- '서울'와 '한국' 사이의 관계와 '도쿄'와 '일본' 사이의 관계가 평행하기 때문에
- '서울' 벡터에서 '한국' 벡터를 빼고 '일본' 벡터를 더하면 그 결과 벡터는 '도쿄' 벡터와 매우 비슷하게 됨.
  - 서울:한국=도쿄:일본. 따라서 서울-한국+일본=도쿄.



Male-Female



Verb tense



Country-Capital

분포 의미론(distributional semantics)

- Word2Vec의 근저에는 분포 의미론(distributional semantics)이 매우 중요한 이론적 전제로서 자리잡고 있음.
- 언어 요소(형태소, 단어 또는 그보다 큰 단위)의 의미는 그 언어 요소의 분포(distribution)에 반영된다는 것.
- 분포란 어떤 언어 요소가 출현하는 언어적 환경/문맥.
  - 어떤 단어 X와 같은 문장 안에서 함께 나타나는 다른 단어들이 X의 분포.
- 의미가 비슷한 언어 요소들은 비슷한 단어들과 함께 나타나는(共起하는, cooccur하는) 경향이 있다는 것.
- Word2Vec에 의해 얻어진 단어 벡터가 단어들 사이의 직관적인 의미적 관계를 잘 나타내므로, 분포 의미론의 전제는 대체로 올바른 방향 위에 있는 것으로 입증되었다고 할 수 있음.

1.3. 벡터들 사이의 유사도

- Word2Vec이나 FSE 알고리즘에 의해 얻어진 벡터는 대개 고차원임. (통상 100차원 정도)
  - 하나의 글자도, 하나의 한시 작품도 [0.234, -0.072, 0.003, ..., -0.014]와 같은 100개의 실수로 이루어진 벡터로 표상됨.
- 글자를 이렇게 벡터로 표상하면, 글자와 글자 사이의 유사도를 계산할 수 있음.
- 유클리드 거리
- (Euclidean distance)

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

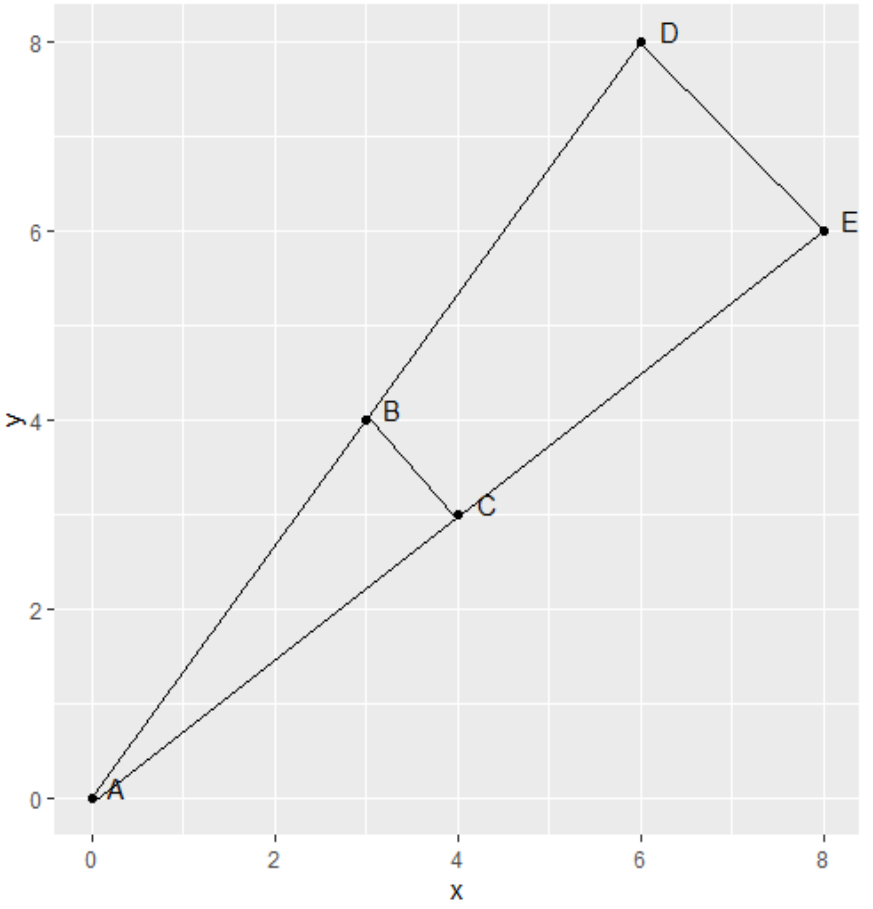
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

- 코사인 유사도
- (cosine similarity)

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

유클리드 거리와 코사인 유사도의 비교

- 유클리드 거리는 2차원 평면상에서 두 점 사이에 선분을 그었을 때, 이 선분의 길이.
  - 거리에 대한 일반인의 직관에 부합.
- 코사인 유사도는, 원점에서 바라보았을 때 두 점 사이의 각도가 작을수록 유사도가 높음.
- B와 C 사이의 유클리드 거리가 B와 D 사이의 유클리드 거리보다 작음.
- 코사인 유사도에 따르면 B와 C보다는 B와 D가 더 유사함.
  - X축을 언어능력, Y축을 수리능력으로 해석하면
  - 전반적인 성적은 B와 C가 비슷하고 B와 D는 큰 차이가 있지만
  - B와 D는 수리능력이 더 우수하고
  - C와 E는 언어능력이 더 우수하다는 공통점이 있음.
  - 코사인 유사도는 방향성을 중시



‘속_01/NNG’와 가장 유사한 10개 요소

순위	단어	유사도	순위	단어	유사도
1	안_01/NNG	0.8127	6	깊숙히/MAG	0.7486
2	물속/NNG	0.7666	7	마음속/NNG	0.7485
3	가슴속/NNG	0.7593	8	꿈속/NNG	0.7485
4	밑바닥/NNG	0.75	9	몸속/NNG	0.7484
5	머리속/NNG	0.7498	10	품안/NNG	0.7412

	기준자	총빈도			기준자	총빈도		
	不	173894			人	133960		
	유클리드	거리	코사인	유사도	유클리드	거리	코사인	유사도
순위	유사자	유사도	유사자	유사도	유사자	유사도	유사자	유사도
1	那	0.7207	那	0.8603	者	0.5750	者	0.7875
2	讵	0.7073	讵	0.8536	鄉	0.5730	鄉	0.7865
3	未	0.6738	未	0.8369	客	0.5636	客	0.7818
4	岂	0.6523	岂	0.8261	娛	0.5544	娛	0.7772
5	也	0.6066	也	0.8033	呸	0.5500	呸	0.7750
6	弗	0.6030	弗	0.8015	酡	0.5490	酡	0.7745
7	锃	0.5971	锃	0.7985	瞅	0.5445	瞅	0.7722
8	盡	0.5936	盡	0.7968	柸	0.5432	柸	0.7716
9	难	0.5926	难	0.7963	撽	0.5329	撽	0.7665
10	宁	0.5886	宁	0.7943	酯	0.5328	酯	0.7664

	기준자	총빈도			기준자	총빈도		
	一	123233			无	107872		
	유클리드	거리	코사인	유사도	유클리드	거리	코사인	유사도
순위	유사자	유사도	유사자	유사도	유사자	유사도	유사자	유사도
1	数	0.6185	数	0.8092	何	0.6343	何	0.8171
2	三	0.5989	三	0.7995	檉	0.5844	檉	0.7922
3	磬	0.5705	磬	0.7853	不	0.5810	不	0.7905
4	几	0.5596	几	0.7798	罣	0.5760	罣	0.7880
5	鞬	0.5488	鞬	0.7744	佯	0.5756	佯	0.7878
6	璔	0.5449	璔	0.7724	癌	0.5737	癌	0.7868
7	筷	0.5417	筷	0.7709	有	0.5713	有	0.7857
8	沏	0.5387	沏	0.7694	壚	0.5692	壚	0.7846
9	葉	0.5363	葉	0.7682	閻	0.5671	閻	0.7836
10	坂	0.5316	坂	0.7658	酡	0.5661	酡	0.7831

	기준자	총빈도			기준자	총빈도		
	风	106742			山	100508		
	유클리드	거리	코사인	유사도	유클리드	거리	코사인	유사도
순위	유사자	유사도	유사자	유사도	유사자	유사도	유사자	유사도
1	飔	0.6267	飔	0.8133	峰	0.6871	峰	0.8435
2	飔	0.6170	飔	0.8085	嶂	0.6550	嶂	0.8275
3	颯	0.5896	颯	0.7948	岩	0.6354	岩	0.8177
4	颸	0.5843	颸	0.7922	岑	0.6206	岑	0.8103
5	吹	0.5700	吹	0.7850	嵋	0.6203	嵋	0.8102
6	凉	0.5657	凉	0.7828	峯	0.6185	峯	0.8092
7	颾	0.5629	颾	0.7815	岭	0.6040	岭	0.8020
8	飔	0.5547	飔	0.7773	峦	0.5980	峦	0.7990
9	飂	0.5340	飂	0.7670	硖	0.5828	硖	0.7914
10	篧	0.5286	篧	0.7643	浭	0.5732	浭	0.7866

	기준자	총빈도			기준자	총빈도		
	有	94072			来	83427		
	유클리드	거리	코사인	유사도	유클리드	거리	코사인	유사도
순위	유사자	유사도	유사자	유사도	유사자	유사도	유사자	유사도
1	在	0.7253	在	0.8626	还	0.6781	还	0.8390
2	是	0.6243	是	0.8121	去	0.6037	去	0.8019
3	讬	0.6131	讬	0.8065	尋	0.5783	尋	0.7891
4	軺	0.6086	軺	0.8043	皈	0.5732	皈	0.7866
5	粹	0.6056	粹	0.8028	浀	0.5579	浀	0.7790
6	驈	0.6040	驈	0.8020	綡	0.5548	綡	0.7774
7	臗	0.6026	臗	0.8013	暖	0.5508	暖	0.7754
8	嶓	0.6022	嶓	0.8011	鳫	0.5470	鳫	0.7735
9	鬩	0.5972	鬩	0.7986	圡	0.5417	圡	0.7708
10	豜	0.5960	豜	0.7980	嗖	0.5401	嗖	0.7700

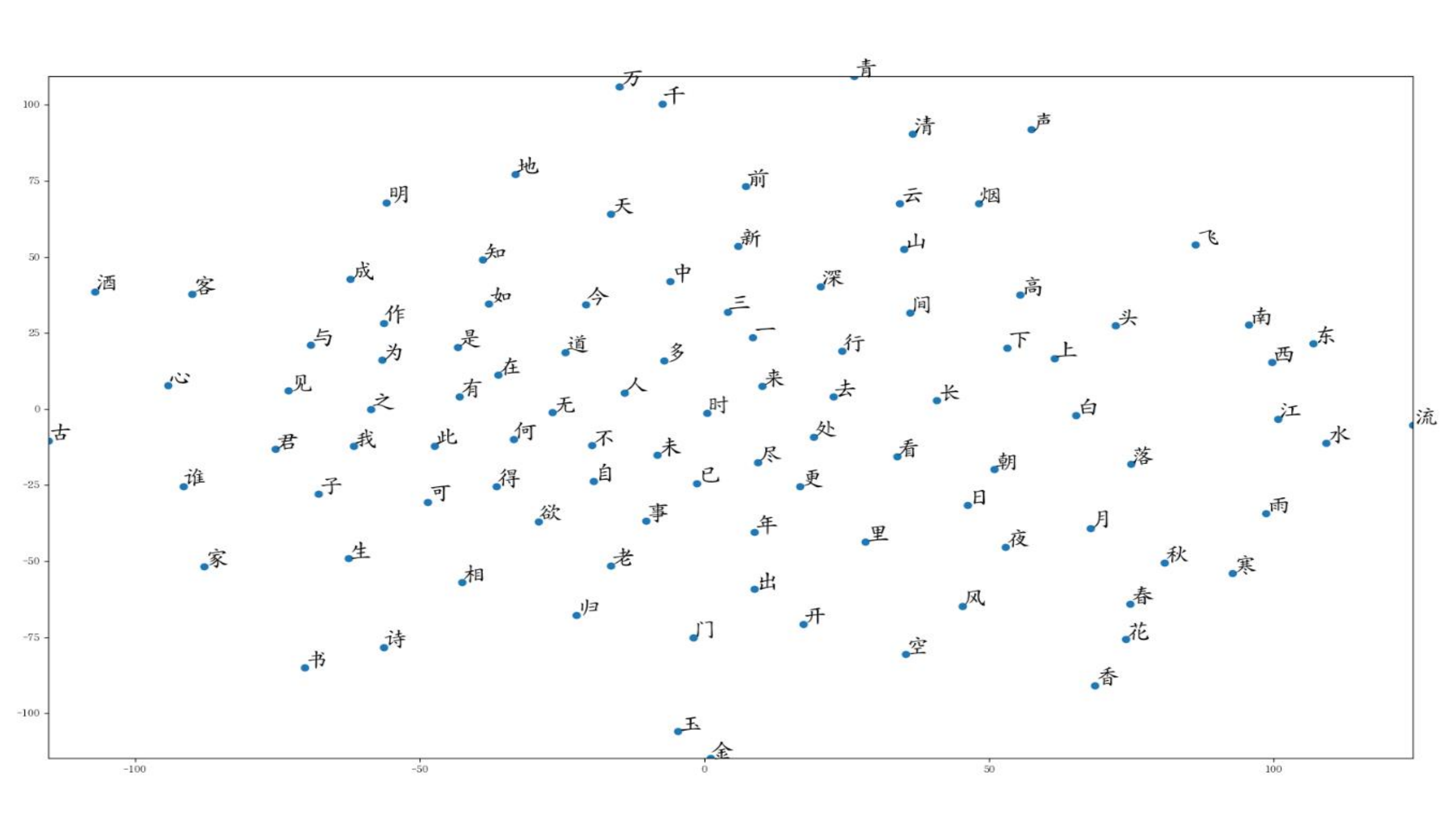
	기준자	총빈도			기준자	총빈도		
	天	81132			日	79090		
	유클리드	거리	코사인	유사도	유클리드	거리	코사인	유사도
순위	유사자	유사도	유사자	유사도	유사자	유사도	유사자	유사도
1	帝	0.5651	帝	0.7826	夕	0.6217	夕	0.8109
2	邽	0.5313	邽	0.7657	旭	0.5859	旭	0.7929
3	穹	0.5309	穹	0.7655	曚	0.5770	曚	0.7885
4	焠	0.5215	焠	0.7607	昳	0.5727	昳	0.7863
5	騱	0.5145	騱	0.7573	眬	0.5670	眬	0.7835
6	霄	0.5101	霄	0.7550	朝	0.5644	朝	0.7822
7	嶊	0.5095	嶊	0.7548	濃	0.5579	濃	0.7789
8	翷	0.5077	翷	0.7538	月	0.5578	月	0.7789
9	极	0.5063	极	0.7531	飁	0.5557	飁	0.7778
10	晏	0.5031	晏	0.7516	旦	0.5550	旦	0.7775

최고 빈도 글자 10개들 사이의 유사도

	不	人	一	无	风	山	有	来	天	日
不	1.0000	0.4109	0.3215	0.5810	0.2050	0.2221	0.3287	0.3183	0.2576	0.3279
人	0.4109	1.0000	0.2341	0.4026	0.3233	0.3507	0.3474	0.3938	0.2537	0.2762
一	0.3215	0.2341	1.0000	0.3431	0.2855	0.2618	0.3200	0.3478	0.2109	0.2135
无	0.5810	0.4026	0.3431	1.0000	0.2080	0.2358	0.5713	0.4172	0.3762	0.2314
风	0.2050	0.3233	0.2855	0.2080	1.0000	0.2563	0.1716	0.3650	0.2176	0.2678
山	0.2221	0.3507	0.2618	0.2358	0.2563	1.0000	0.2502	0.3325	0.3488	0.3086
有	0.3287	0.3474	0.3200	0.5713	0.1716	0.2502	1.0000	0.3388	0.2866	0.1687
来	0.3183	0.3938	0.3478	0.4172	0.3650	0.3325	0.3388	1.0000	0.3496	0.2805
天	0.2576	0.2537	0.2109	0.3762	0.2176	0.3488	0.2866	0.3496	1.0000	0.3299
日	0.3279	0.2762	0.2135	0.2314	0.2678	0.3086	0.1687	0.2805	0.3299	1.0000

차원 축소(dimensionality reduction)

- Word2Vec 알고리즘을 통해 얻어진 100차원의 수치 벡터는 그 자체로서는 직관적으로 이해하기가 어려움.
- 차원축소법(dimensionality reduction)을 이용하여 2차원으로 축소하고 이를 2차원 평면상에 그래프로 나타내면 직관적으로 이해하기가 쉬워짐.
- 최고 빈도 글자 100개의 그래프: 다음 슬라이드
- 의미상 서로 관련된 유사한 글자들이 가까이에 위치해 있는 것을 볼 수 있음.


万
千
青
清
声
地
前
明
天
云
烟
新
山
飞
知
成
中
深
酒
客
高
如
今
三
间
头
南
作
一
东
与
是
道
多
行
下
上
西
为
在
心
见
有
人
来
去
长
之
无
时
白
江
流
古
君
我
此
何
不
未
处
看
水
尽
朝
落
谁
子
可
得
自
已
更
日
欲
事
雨
年
月
里
夜
家
生
老
秋
寒
相
出
风
春
归
开
门
诗
空
花
书
香
玉
金
100
75
50
25
0
-25
-50
-75
-100
-100
-50
0
50
100

1.4. 단어벡터/글자벡터로부터 문장벡터로

- 실용적인 언어 처리 task 가운데에는 단어보다 훨씬 큰 언어단위(문장, 문서 등)를 대상으로 한 것들이 많이 있음.
  - 예: 문서 분류, 문서 요약, 문서/문장간 유사성/관련성 측정, 기계 독해, 기계 추론, 기계 번역
- 이를 위해서는 단어보다 큰 언어단위(문장, 문서, 작품 등)도 벡터화해야 함.
- 상식적인 방법은, 이미 구해 놓은 단어벡터들을 바탕으로 하여 문장벡터를 구성하는 것임.
- 단어벡터들을 어떻게 종합하여 문장벡터를 구성할 것인가?

벡터의 평균

- 학급의 修學 능력을 측정하는 경우
  - 학급에 속한 학생의 벡터(과목별 점수 목록)들을 모두 (과목 점수별로) 더한 뒤 학급 내 학생 수로 나눔. 즉 평균 벡터를 구함.
  - 이렇게 얻어진 학급 벡터는 학생들의 학업 성향을 종합적으로 반영함.
    - 수학을 잘 하는 학생들이 많이 몰려 있는 학급은 수학 평균 점수가 높음.
- 단어벡터로부터 문장벡터를 만드는 경우
  - 문장에 포함된 단어들의 벡터를 모두 (요소별로) 더한 뒤 단어 수로 나눔. 즉 평균 벡터를 구함.
  - 이렇게 얻어진 문장벡터는 단어들의 성향/의미를 종합적으로 반영함.
    - 야구에 관한 단어들이 많이 포함되어 있는 문장은 야구에 관한 문장일 확률이 높음.

가중평균

- 야구선수 벡터로부터 야구팀 벡터를 만드는 경우
  - 팀내의 모든 선수를 똑같이 취급하여 평균 벡터를 구하는 것은 온당치 않음.
  - 각 선수의 중요도를 측정한 뒤, 중요도/가중치를 반영할 필요가 있음.
  - 즉 가중평균(weighted average)/가중합(weighted sum)을 구해야 함.
  - 각 가중치는 0과 1 사이의 실수, 가중치의 합은 1임.
  - 예: 0.5×류현진 + 0.4×벨린저 + 0.1×후보선수
- 단어벡터로부터 문장벡터를 만드는 경우
  - 문장 내의 모든 단어의 중요도가 똑같지는 않음.
  - 조사, 어미, 기능어 등의 초고빈도어는 주제/영역에 상관없이 어느 문장에나 나타날 수 있음.
  - 그런 요소보다는 특정 주제/영역에 유독 많이 나타나는 단어가 더 중요함.
  - 단어 가중평균의 예: 유격수가 1루로 던졌습니다.
    - 0.35×유격수 + 0.02×가 + 0.35×1루 + 0.04×로 + 0.2×던지 + 0.02×었 + 0.02×습니다

가중평균을 이용한 문장 벡터화: FSE

- Arora, Liang and Ma (2017): Fast Sentence Embedding (FSE)
  - ①문장 내 각 단어를 Word2Vec 알고리즘으로 벡터화.
  - ②문장 내 각 단어의 SIF(smooth inverse frequency)를 단어 벡터의 가중치로 곱하여 평균 벡터를 구함.
    - 말뭉치에서 빈도가 높을수록 SIF는 낮게 되어 있음.
  - ③이렇게 얻어진 문장 벡터에서 C_0(가장 크기가 큰 차원) 성분을 뺌.
    - C0는 주로 주제/영역과 상관없이 모든 문서에 얼마든지 나타날 수 있는 초고빈도어, 기능어, 문법요소에 해당.
- FSE는 문장을 단어 벡터들의 가중평균으로 종합하여 나타내되, 같거나 비슷한 단어들이 많이 나타날수록 문장 벡터도 유사하게끔 되어 있음.
- 이 방법은 신경망을 이용하여 언어 모델을 훈련시키는 기타 문장 벡터화 방법들에 비해 훨씬 간단함에도 불구하고, 여러 downstream task에서 매우 우수한 성능을 보임.
- Word2Vec의 아이디어를 그대로 문서에 적용한 Doc2Vec 알고리즘도 시도해 보았으나, 동형어/다의어 실험에서는 FSE에 비해 성능이 낮음.

동형어와 다의어

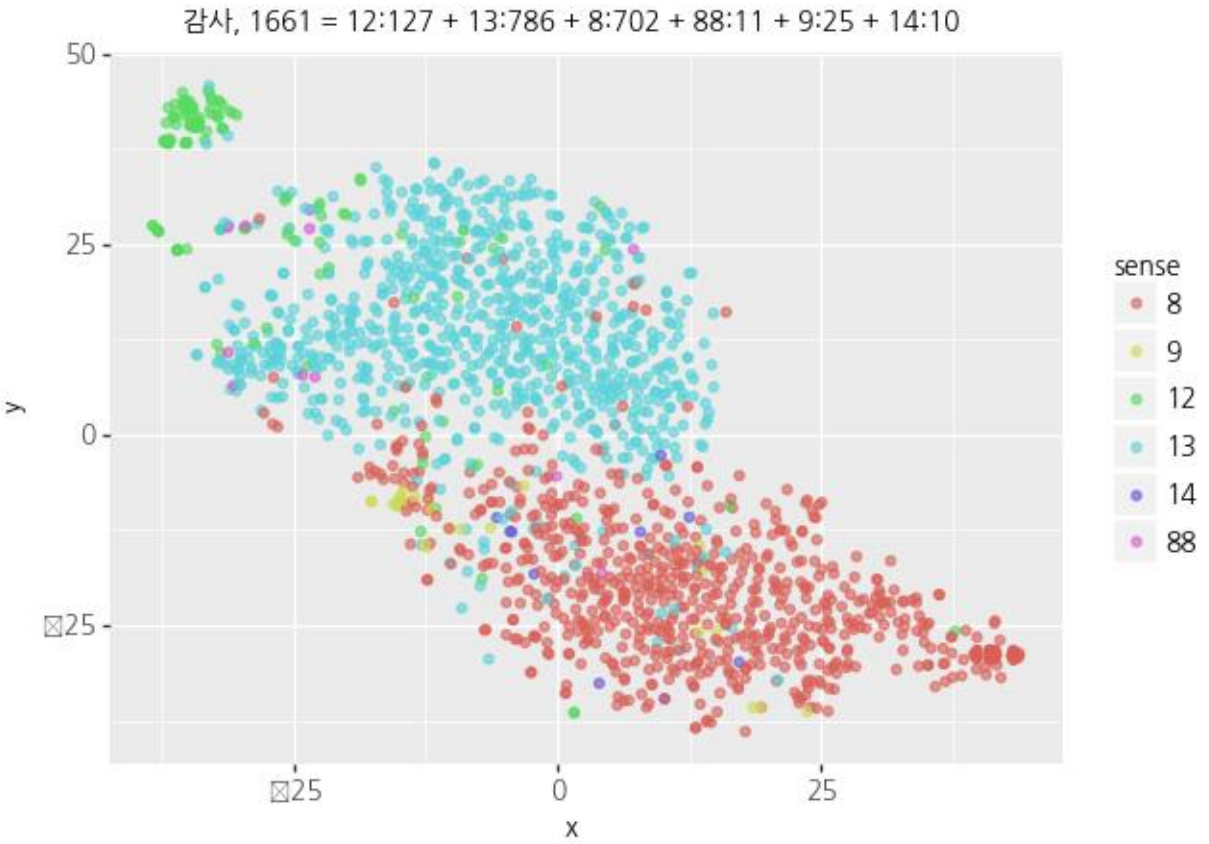
- 동형어(homonym): 둘 이상의 단어가 우연히 형식(form)이 같은 경우
  - 예: '다리1'(leg)과 '다리2'(bridge)
  - 동음어(homophone): 발음이 같은 경우. 예: 'see'와 'sea'
  - 동철어(homograph): 철자가 같은 경우. 예: 'bow'(활)와 'bow'(절하다)
  - 동철동음어: 철자도 같고 발음도 같은 경우. 예: '은행'(銀行)과 '은행'(銀杏)
- 다의어(polysemous word): 둘 이상의 의항(義項, sense)을 갖는 단어
  - 예: '다리1': ①사람/동물의 leg, ②물체의 아래 부분, ③오징어/문어의 leg, ④안경의 부분(귀에 거는 부분)
- 동형어 중의성 해소(homonymy disambiguation)
  - input 단어가 동형어들 중 어느 것인지 판별. (제2절 실험 주제)
- 다의어 중의성 해소(polysemy disambiguation) = WSD(word sense disambiguation)
  - input 단어가 해당 문장에서 어느 의항(sense)으로 쓰였는지 판별. (제3절 실험 주제)

2. 동형어 구분
실험1 절차

- ① Word2Vec을 이용한 단어 벡터화.
  - 세종 형태의미 분석 말뭉치를 문장 단위로 분리.
  - 토큰: 형태(__동형어번호)/품사
  - 형태가 같아도 품사나 동형어 번호가 다르면 서로 다른 토큰으로 간주됨.
- ② FSE를 이용한 문장 벡터화.
- ③ 세종 형태의미분석 말뭉치에서 고빈도 동형어 명사 285개 추출.
  - 동형어들 전체의 빈도가 높을 뿐 아니라, 동형어 각각도 고빈도인 것.
- ④ 각 명사가 출현하는 예문 추출. (예문과 동형어 번호 pairing)
- ⑤ 이 예문들을 FSE 알고리즘에 다시 입력으로 넣어 각 예문의 벡터 표상을 얻음.
- ⑥ 동형어의 예문 벡터들을 (t-SNE 알고리즘으로) 2차원으로 축소하여 2차원 평면에 plotting함. (동형어 번호에 따라 서로 다른 색으로 표시)

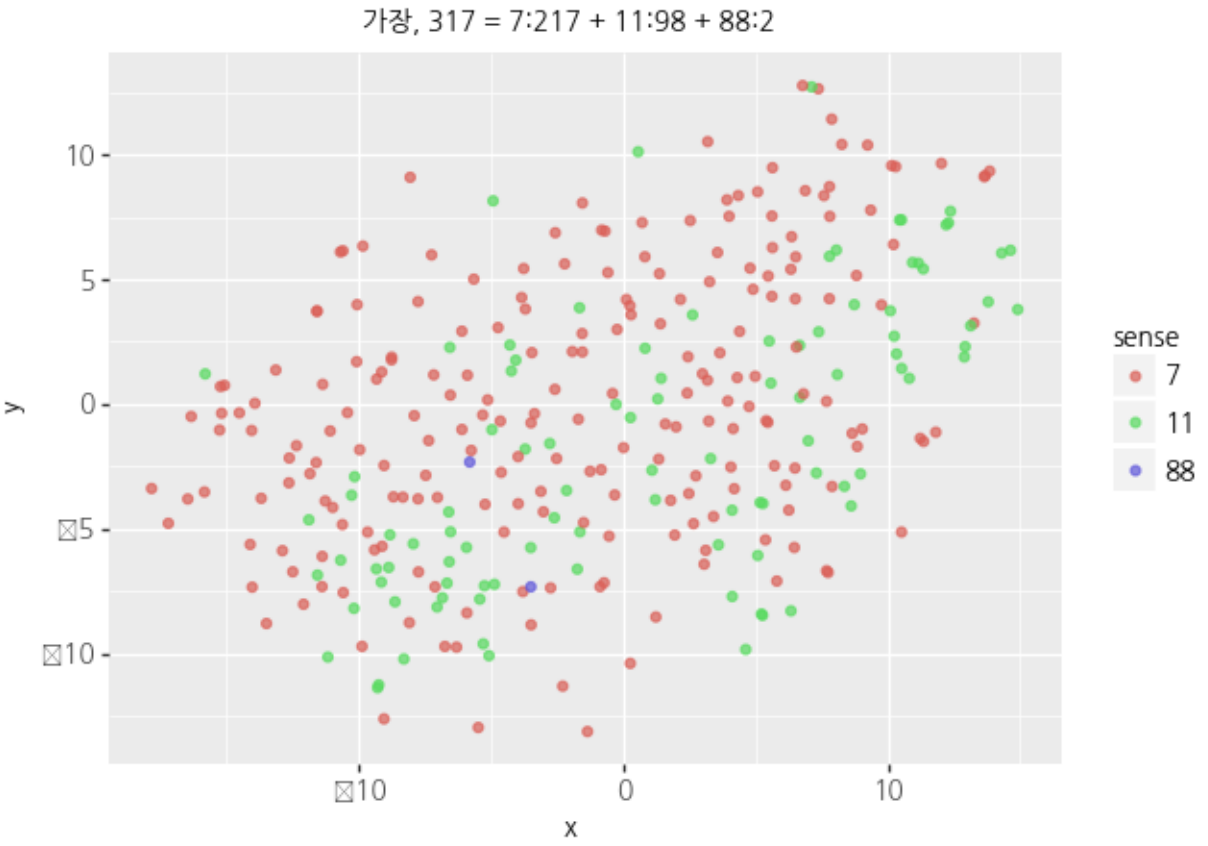
통상적인 벡터화 방법의 결과: 잘 된 경우

- 285개 명사를 모두 plotting 했으나 여기서는 하나만 예시함.
- '감사'의 plot을 보면
- '감사08(感謝, 적색)', '감사13(監査, 민트색)', '감사12(監事, 연두색)'가 비교적 명확히 구분되어 있음.
- 단어벡터와 문장벡터는 서로 비슷한 것들이 비슷하게 벡터화되는 것도 중요하지만
- 서로 다른 것들이 다르게 벡터화되는 것도 중요함.
- '감사'의 plot은 문장벡터가 이 두 덕목을 갖추고 있음을 보여줌.



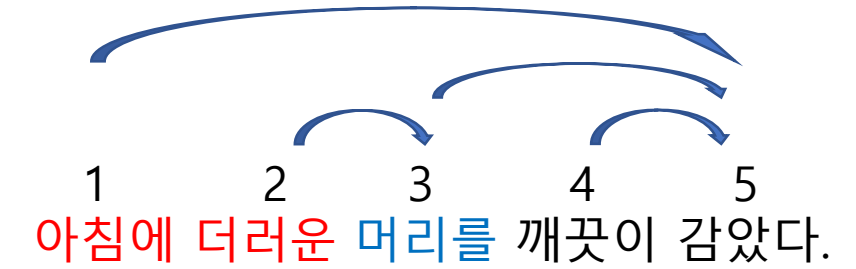
통상적인 벡터화 방법의 결과: 잘 안 된 경우

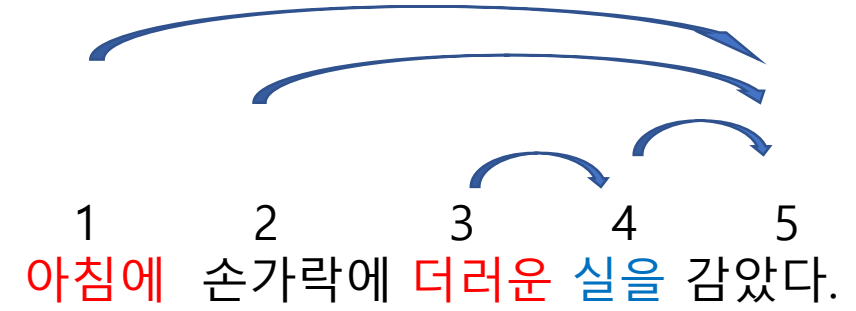
- 반면에 '가장'의 plot을 보면
- '가장07(家長, 적색)'과 '가장11(假裝, 연두색)'이 뒤섞여 있음.
- 이 결과를 곧이곧대로 받아들여 해석하자면, 이 두 개의 '가장'은 분포상 별다른 차이를 보이지 않는다고 결론짓게 됨.
- 그러나 동형어가 더 명확히 구별되도록(분포 차이를 더 잘 포착하도록) 벡터화 방법을 개선할 수도 있지 않을까?

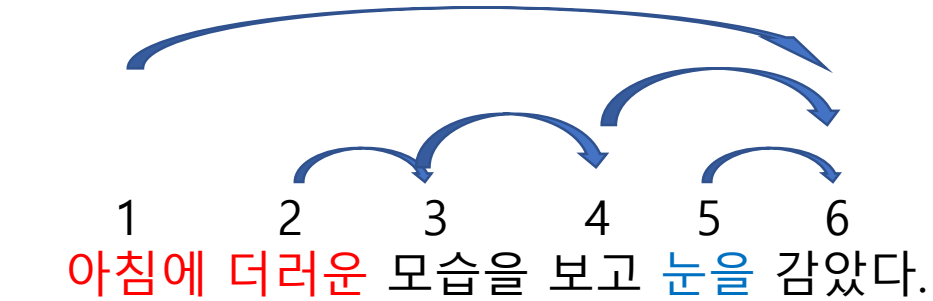


문장 벡터에서 통사적 의존관계의 중요성

- 동형어 구분의 타깃이 동사 '감다'인 경우
- '감다1'(눈을 감다)은 '눈'을 목적어로 할 때가 많고
- '감다2'(머리 감다)는 '머리'를 목적어로 할 때가 많고
- '감다3'(빙 두르다)은 '실, 테이프, 붕대' 등을 목적어로 할 때가 많음.
- 타깃 단어와 밀접한 통사 관계를 맺는 요소는 타깃 단어의 구별에 도움이 되지만
- 그 외의 요소(수식어의 수식어, 시공간 부사어 등)는 오히려 방해가 될 수 있음.







통사적 의존관계의 중요성

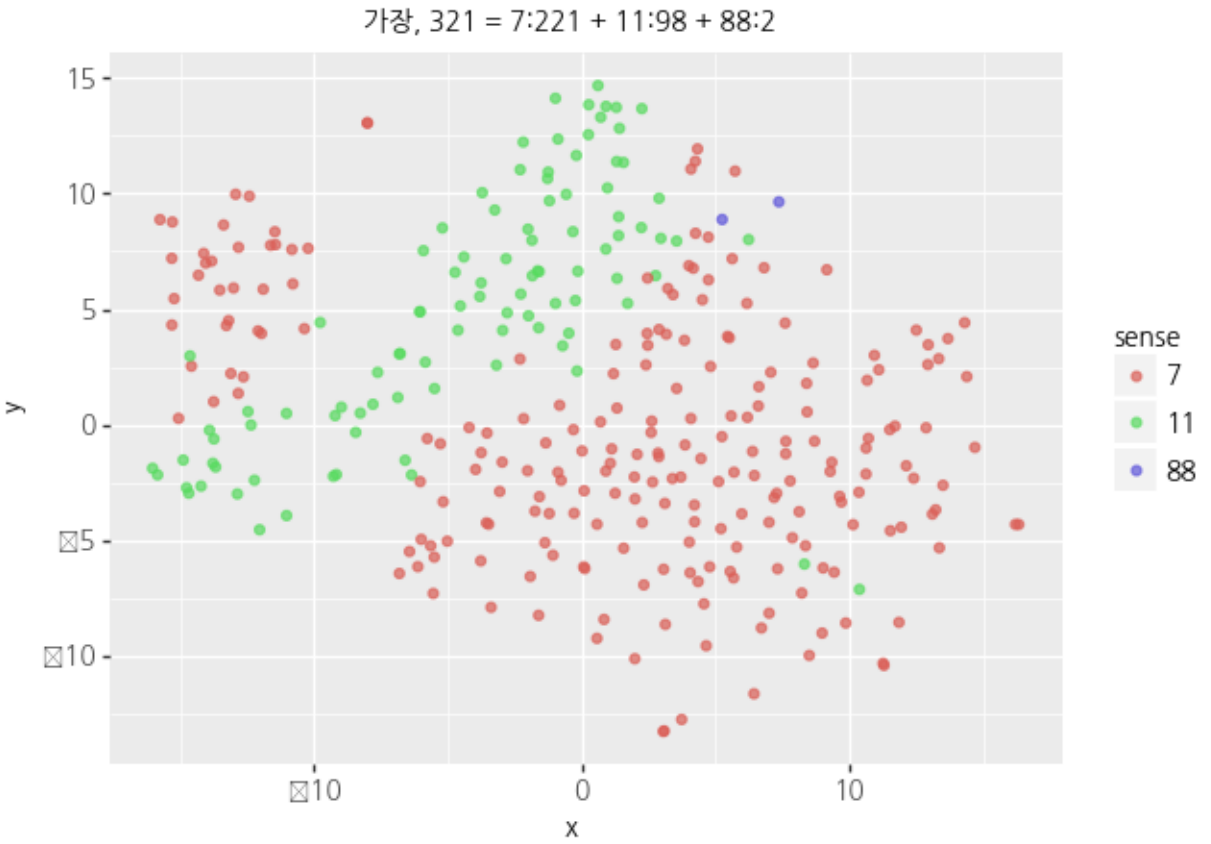
- 타깃 단어가 명사인 경우에도 마찬가지임.
  - (4) 아침에 큰 감나무에서 감을 따서 먹었다.
  - (5) 아침에 시장에 가서 큰 이불을 만들 감을 샀다.
  - (6) 아침에는 오늘 큰 거 한 건 할 것 같은 감이 들었는데, 실제로는 그렇지 않았다.
- '감1'(과일)은 '먹다, 맛있다' 등과 잘 결합하고
- '감2'(재료)는 '사다, 만들다, 부족하다, 충분하다' 등과 잘 결합하고
- '감12'(感)는 '들다, 있다, 좋다, 떨어지다' 등과 잘 결합함.
- 즉 '감'과 직접 통사 관계를 맺는 요소들은 동형어 구별에 도움이 됨.
- 반면에 '아침에'처럼 타깃 단어와 직접 통사 관계를 맺지 않는 요소는 오히려 (차이를 흐려서) 방해가 될 수 있음.
- 타깃 단어의 개성을 잘 반영하는 요소만 선별하여 벡터화할 필요가 있음.

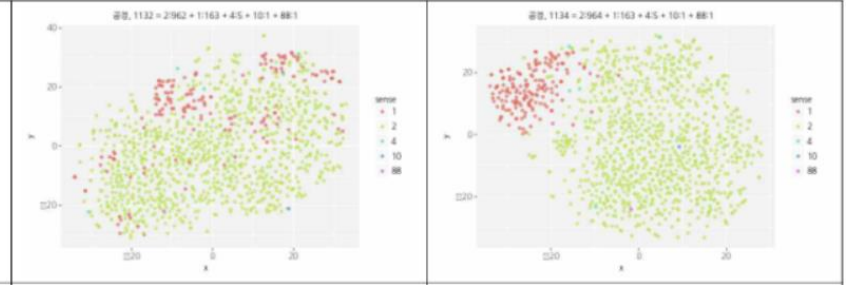
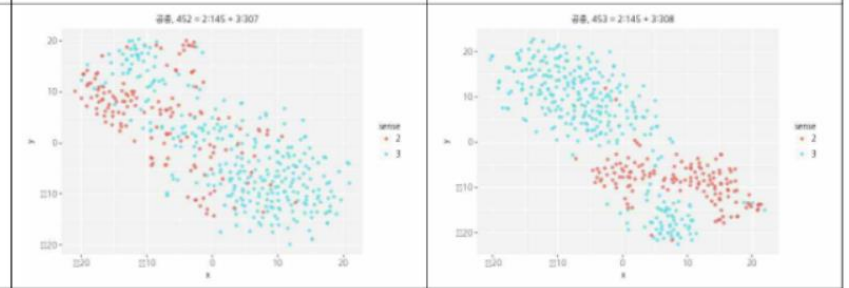
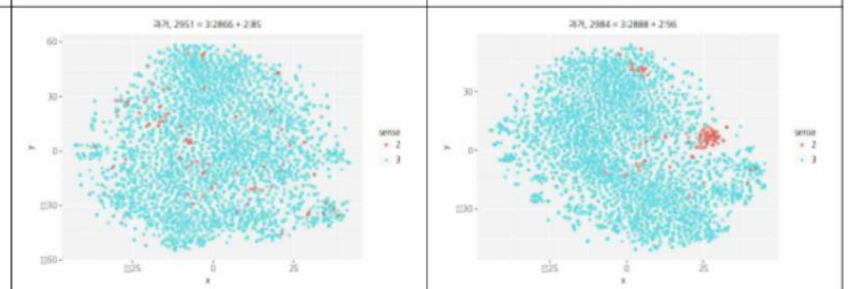
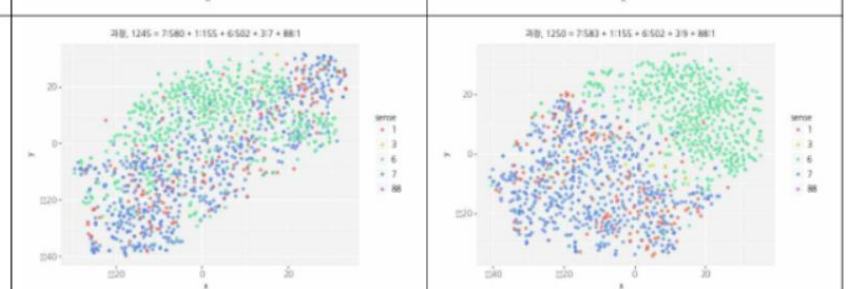
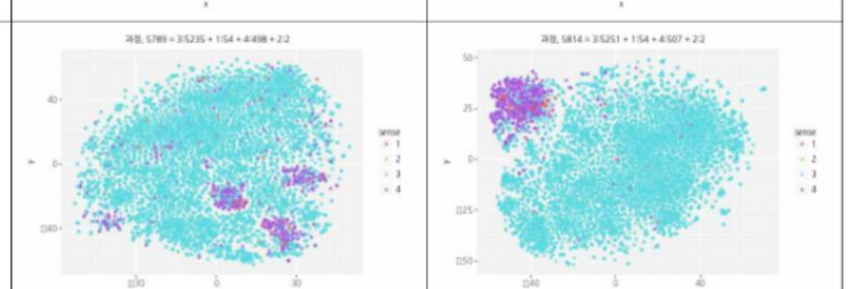
동형어 구분 실험2 절차

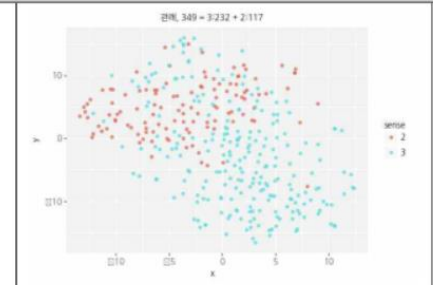
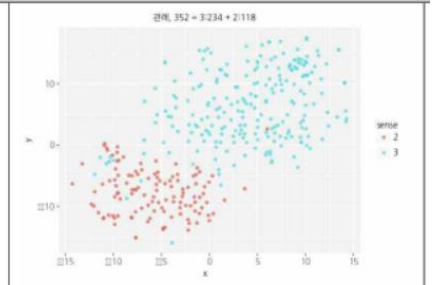
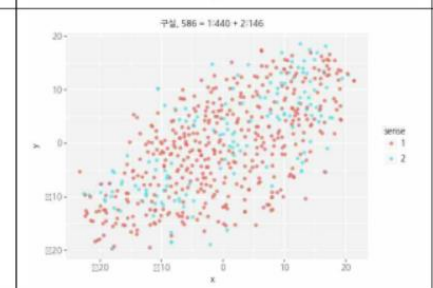
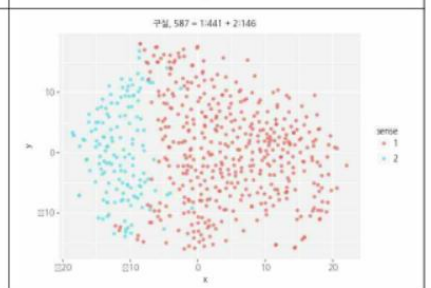
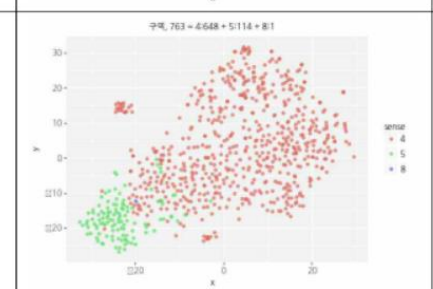
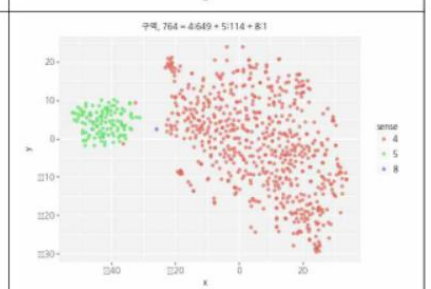
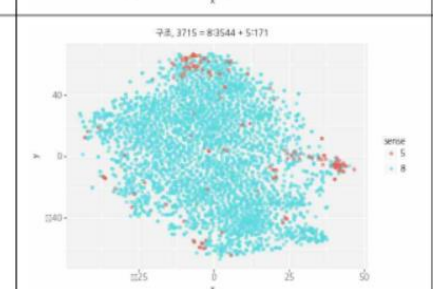
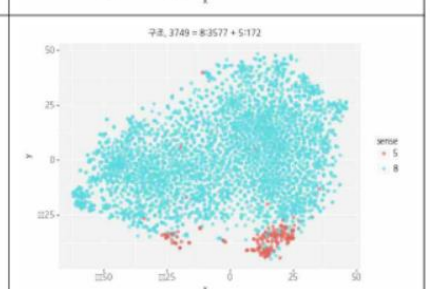
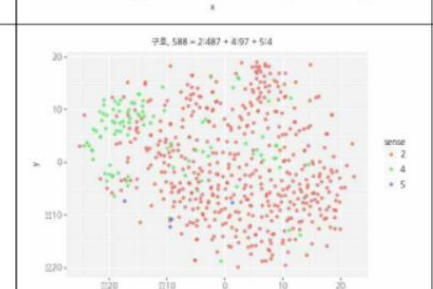
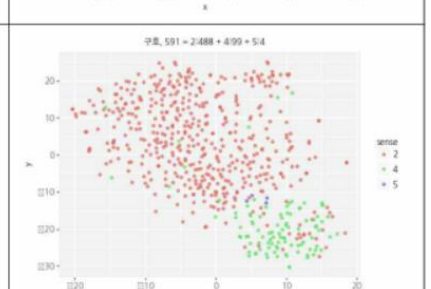
- ⓐ 구문분석(parsing)
  - 자동 구문분석기(parser)로 세종 형태의미분석 말뭉치를 구문분석.
  - 박진호(2003)에 소개된 구문분석기 이용.
- ⓑ 실험1의 ①~④와 동일한 절차를 밟되, 어절을 선별함.
  - 타깃 명사가 출현한 예문에서 다음의 3가지 어절들만 선별.
  - (1) 해당 명사를 포함한 어절
  - (2) 이 어절을 지배하는 어절,
  - (3) 이 어절이 지배하는 의존소 어절
- ⓒ 실험1의 ⑤~⑥과 동일한 절차를 밟아, 동형어 plot을 얻음.

실험2 결과 예시

- 285개 명사 중 일부의 plot은 다음 슬라이드 참조.
- '가장'의 plot을 보면
- '가장07(家長, 적색)'과 '가장11(假裝, 연두색)'이 매우 뚜렷하게 나뉘어 있음.
- 문장 전체를 벡터화했을 때는 동형어들의 벡터가 잘 구분되지 않았던 데 비해
- 문장 내에서 타깃 단어와 직접 통사 관계를 맺고 있는 어절들만 선별해서 벡터화하니, 동형어가 잘 구분됨.



35	공정		
36	공중		
37	과거		
38	과장		
39	과정		

40	관례		
41	구실		
42	구역		
43	구조		
44	구호		

실험2 결과

- 285개 명사 전체의 실험 결과를 종합하면, 통사 관계를 반영하여 벡터화했을 때 (그렇지 않았을 때에 비해) 동형어들을 훨씬 더 뚜렷하게 나누는 경향이 매우 강함.
  - 앞 슬라이드의 좌측 plot에 비해 우측 plot이 향상된(동형어가 더 뚜렷하게 나뉜) 경우가 대부분임.
  - 두 plot이 별 차이가 없는 경우도 소수 있음. 예: 강화, 개정, 규정, 사채, 성적, 세계
  - 좌측 plot에 비해 우측 plot이 더 악화된(동형어가 더 뒤섞인) 경우는 없음.
- 야구 팀에서 별 특징 없는 그저그런 선수들을 제외하고 팀 컬러를 대표하는 선수들만 추리면, 해당 팀의 팀 컬러가 강화되듯이
- 문장에서 별 특징 없는 단어들을 제외히고, 타깃 단어의 특징을 잘 보여주는 단어들만 추리면, 해당 문장의 특징이 더 뚜렷해짐.
- 타깃 단어의 특징을 잘 보여주는 단어는, 타깃 단어와 통사적으로 밀접한 관계를 맺고 있는 단어임.

3. 다의어 의항 구분

- 이 절에서는, 타깃 단어가 포함된 문장들을 벡터화한 뒤, 다차원 벡터 공간에서 이 벡터들을 군집화(clustering)함으로써 WSD(다의어의 의항 구별)의 문제에 접근.
- 2303개의 용언에 대한 실험 결과 소개.
- 동형관계에 있는 두 단어를 구별하는 것보다 다의어의 의항을 구별하는 것이 난이도가 더 높음.
- 하지만 분포 의미론의 가설에 따르면, 동형어뿐 아니라 다의어의 별개의 의항들도 분포상의 차이로 반영될 터.
- 동형어의 경우, 정답[ground truth]이 달려 있는 자료가 있으므로 기계학습 기법 중 지도학습(supervised learning) 기법을 이용할 수 있으나
- 다의어의 경우 그런 자료가 없으므로, 비지도학습 기법을 이용해야 함.

실험 절차

- ① 말뭉치 전처리: 세종 형태의미분석 말뭉치, 모두의 말뭉치
- ② 토큰(단어, 형태소 포함) 벡터화: Word2Vec
- ③ 구문분석: 박진호(2003)의 구문분석기
- ④ 타깃 단어 선정: 2303개 다의 용언
- ⑤ 문장 벡터화: FSE
- ⑥ 군집화: K-Means, GMM, BGM
- ⑦ 차원 축소와 시각화: t-SNE, Plotnine

군집화 방법1: K-Means

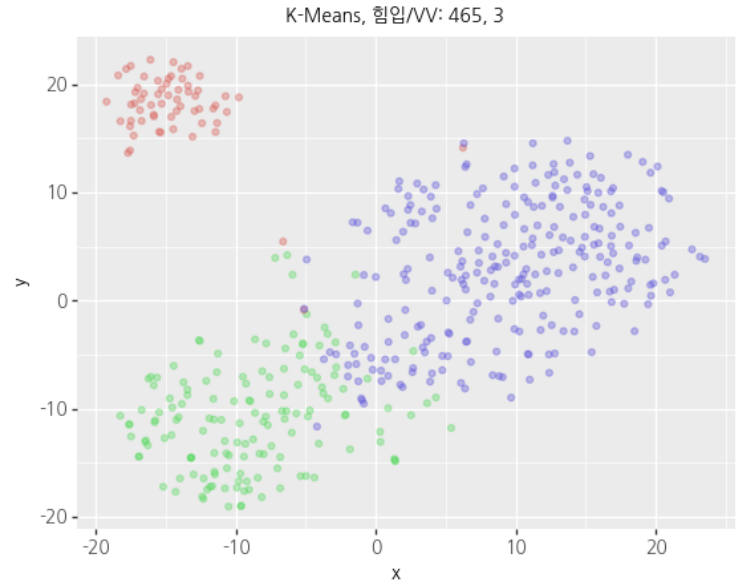
- K-Means Clustering 알고리즘
  - ① 주어진 군집의 개수 k에 따라, 입력 관측점(벡터)들 가운데 랜덤하게 k를 뽑아, 이를 각 군집의 중심(centroid)으로 삼음.
    - 이때 각 중심들이 벡터 공간에서 한 곳에 몰려 있지 않고 가급적 떨어져 있게 함.
  - ② 각 관측점을, k개의 중심 중 거리가 가장 가까운 것의 군집에 소속시킴.
  - ③ 모든 관측점의 소속이 결정되면, 각 군집의 중심을 새로 산출.
  - ④ ②와 ③의 과정을 (각 군집의 중심이 더 이상 유의미하게 이동하지 않게 될 때까지) 반복.
- 이 알고리즘의 1회 시행으로 global optimum을 얻는다는 보장은 없으나
- 여러 차례 반복하여 가장 점수가 좋은 solution을 선택.
- 각 관측점과 군집 중심점 사이의 거리를 바탕으로 하기 때문에, 결과로 얻어진 군집은 球形(spherical)의 모습을 띰.
- 각 군집의 크기(포함하는 관측점의 수)가 대체로 엇비슷하게 됨.
- 이 두 성질이 실제 데이터와 부합되지 않을 수 있음.

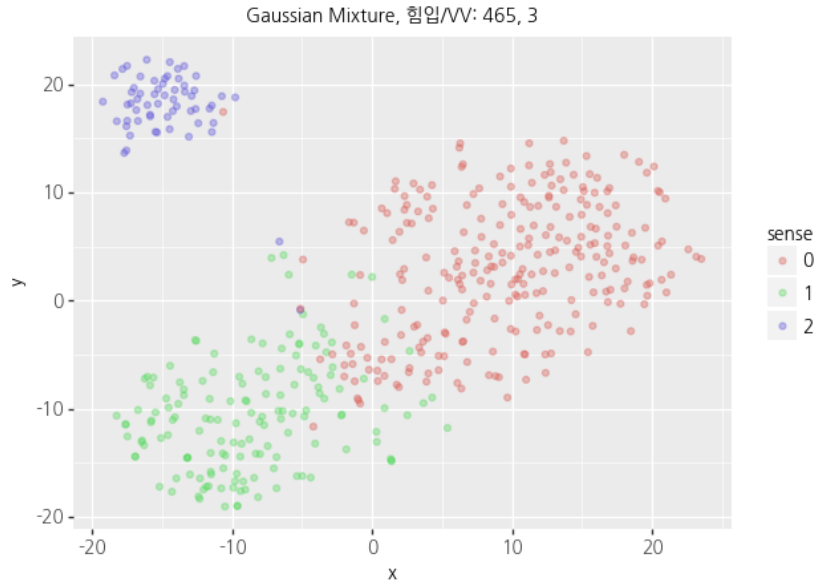
군집화 방법2: Gaussian Mixture Model (GMM)

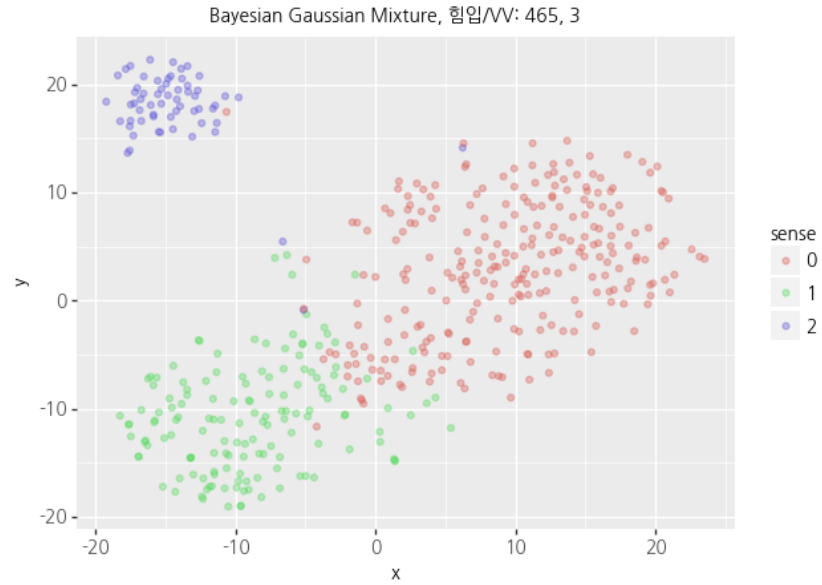
- 입력 데이터가 각각 몇 개(주어진 k개)의 다차원 정규분포를 따른다고 가정.
- 이 k개 정규분포의 모수(평균, 공분산 행렬)를 알아내려는 방식으로 작동.
- 평균은 군집의 위치를 나타내고
- 공분산 행렬은 군집의 모양(구형에 가까운지 아니면 얼마나 길쭉한 타원체인지)과 방향(좌표축과의 각도)을 나타냄.
- 결과로 얻어진 각 군집은 구형일 수도 있지만 길쭉한 타원체 모습을 띨 수도 있고, 이러한 길쭉한 타원체의 방향도 다양할 수 있음.
  - 이는 K-Means를 일반화한 것이라고 할 수 있음.
  - 구형뿐 아니라 훨씬 더 다양한 형태의 군집을 찾아낼 수 있기 때문.
- K-Means에 비해 훨씬 시간이 많이 걸림.
  - 관측점의 수가 많고 벡터의 차원이 높을수록 소요 시간은 엄청나게 증가

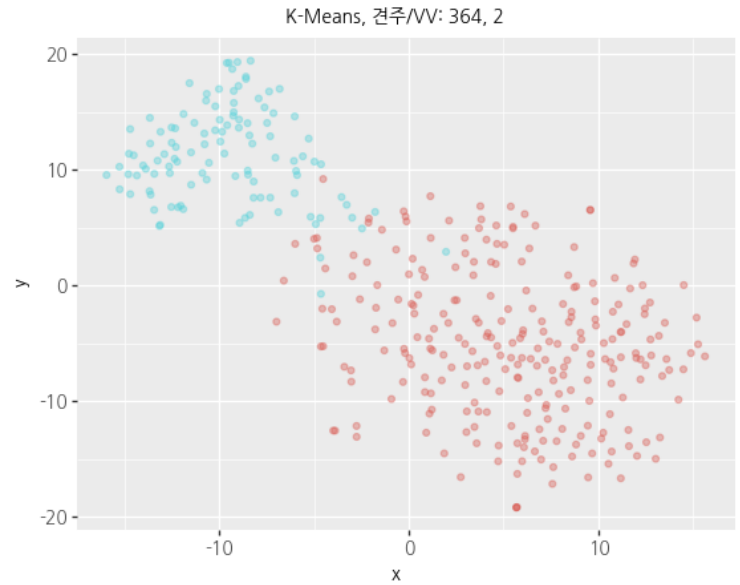
군집화 방법3: Bayesian Gaussian Mixture Model

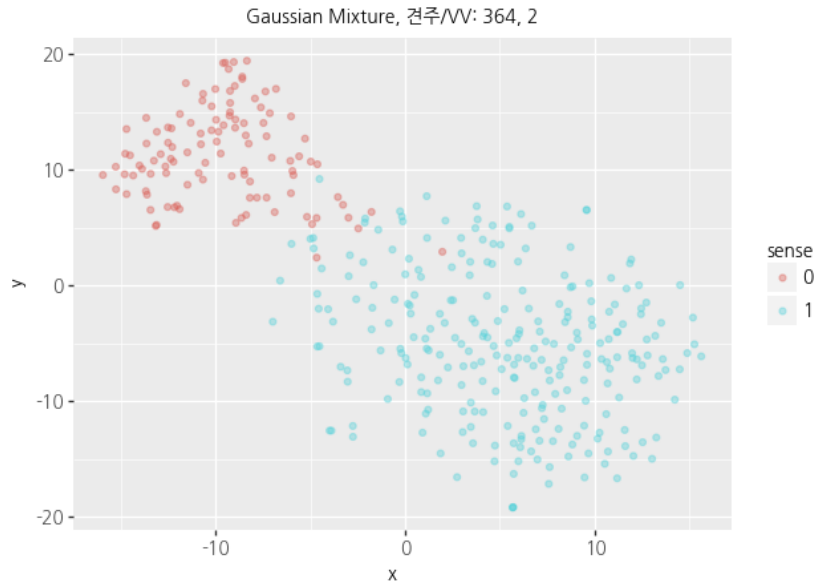
- 관측점을 하나하나 k개의 군집 중 하나에 할당하되
- 이미 형성된 군집들 중 크기가 큰 군집일수록 새로운 관측점을 자기에게 끌어올 확률이 더 높음. (富益富 현상)
- 초기에 관측점을 끌어오지 못한 군집은 최종적으로 매우 적은 수의 관측점을 갖게 되며, 심지어는 관측점이 0개일 수도 있음.
  - 즉 k개라는 군집의 수가 알고리즘에 주어지기는 하나
  - 결과로 얻어지는 군집의 수는 k에 미달할 수도 있음.
- 군집의 크기가 들쭉날쭉하고, 군집의 수가 몇 개인지 사전에 미리 알기가 어려울 때는 이 방법이 제격.
  - 미리 정하는 k값이 결정적 역할을 하는 K-Means나 GMM에 비해 장점을 지님.
- 2303개 용언에 대해 K-Means, GMM, BGM의 세 기법으로 군집화 실시.
  - 단, 빈도가 2만회가 넘는 22개 용언은 연산자원과 시간 문제로 K-Means만 실시.

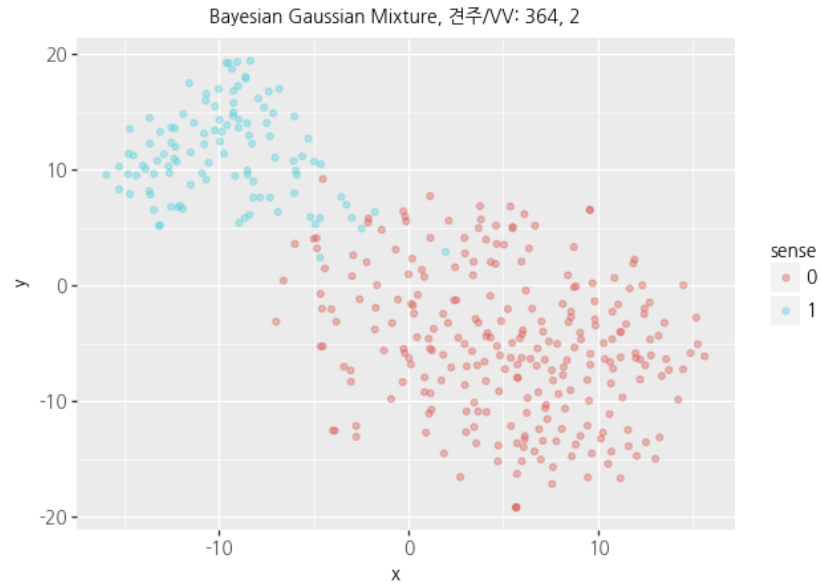

K-Means, 힘입/VV: 465, 3
sense
0
1
2
x
y

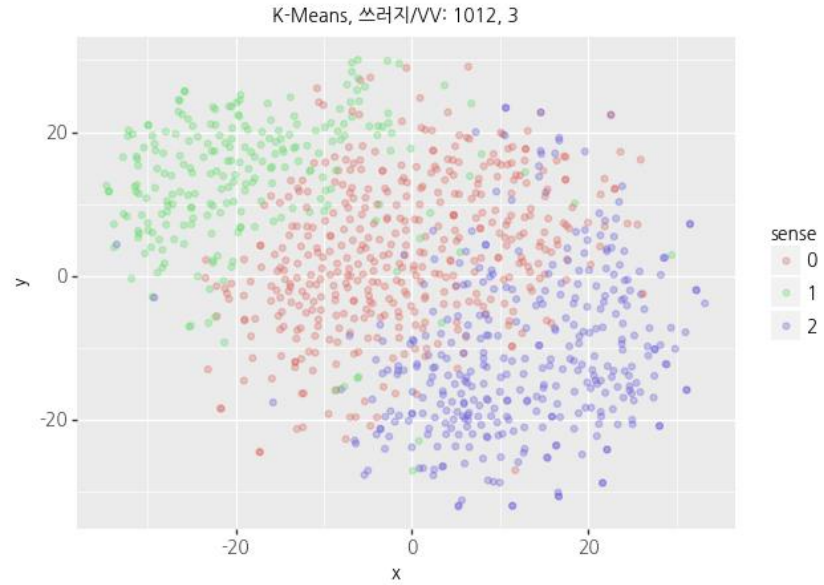

Gaussian Mixture, 힘입/VV: 465, 3
sense
0
1
2
x
y

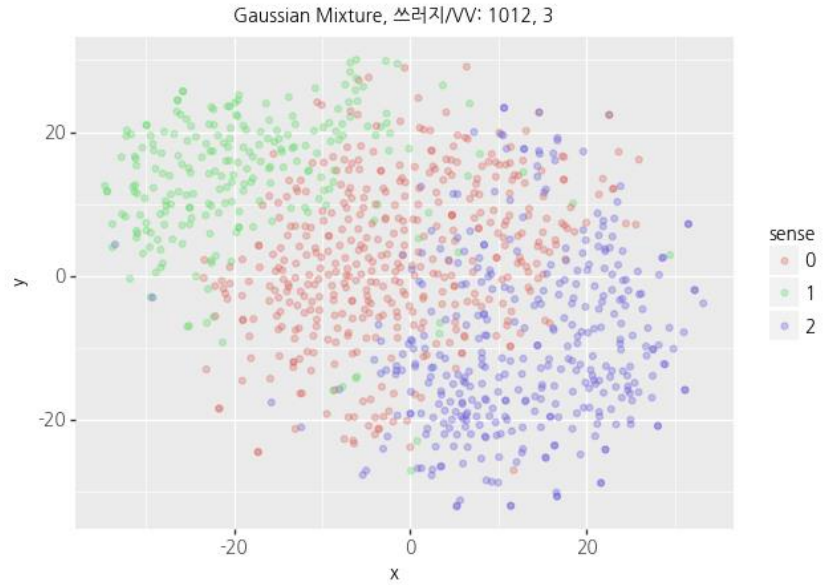

Bayesian Gaussian Mixture, 힘입/VV: 465, 3
sense
0
1
2
x
y

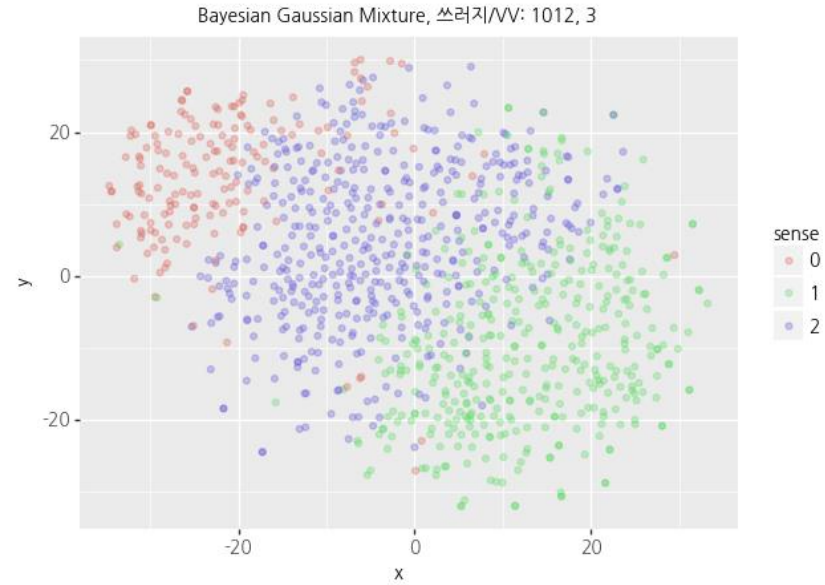

K-Means, 견주/VV: 364, 2
sense
0
1
x
y

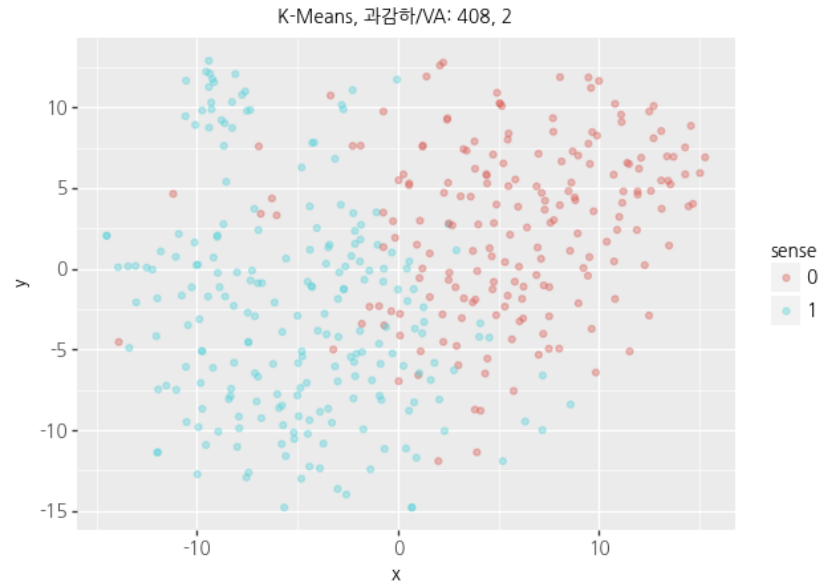

Gaussian Mixture, 견주/VV: 364, 2
sense
0
1
x
y

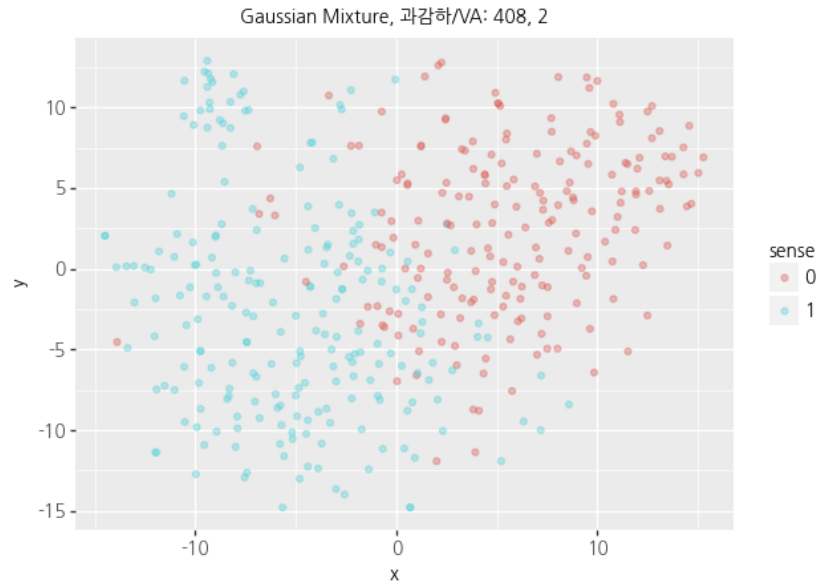

Bayesian Gaussian Mixture, 견주/VV: 364, 2
sense
0
1
x
y

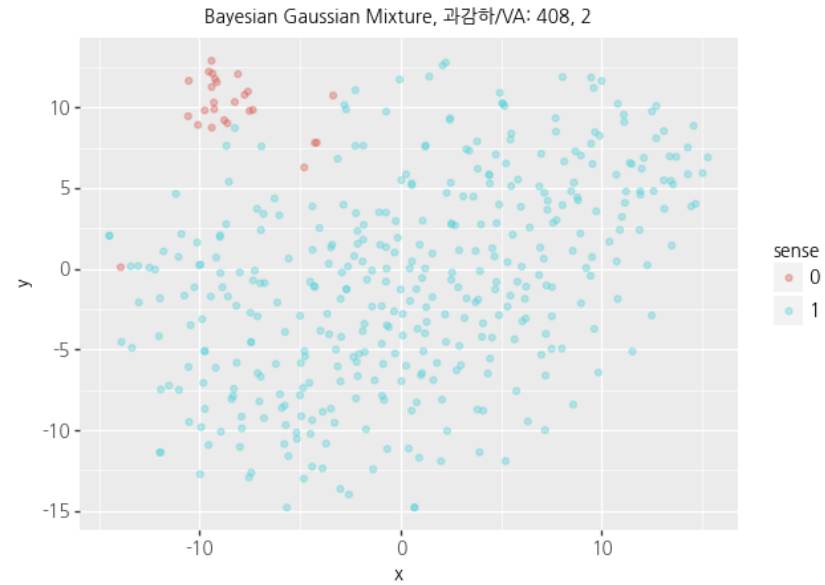

K-Means, 쓰러지/VV: 1012, 3
sense
0
1
2
x
y

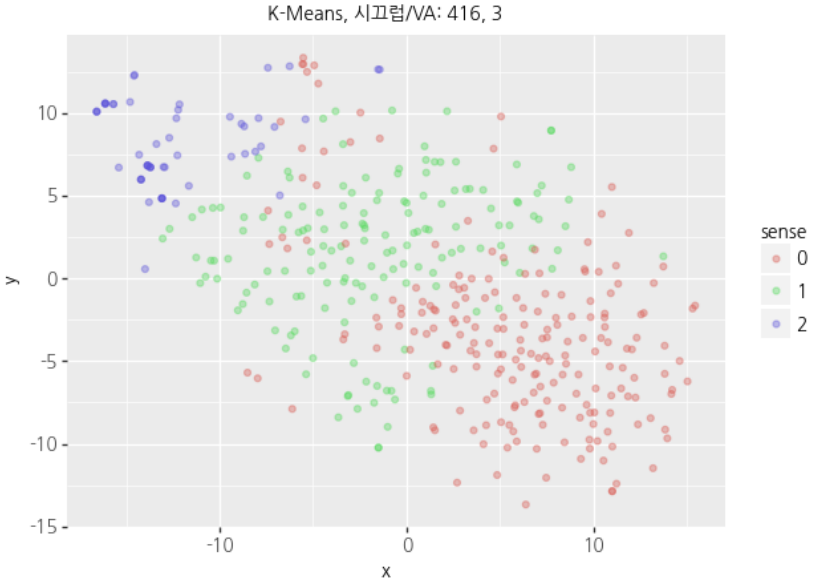

Gaussian Mixture, 쓰러지/VV: 1012, 3
sense
0
1
2
x
y

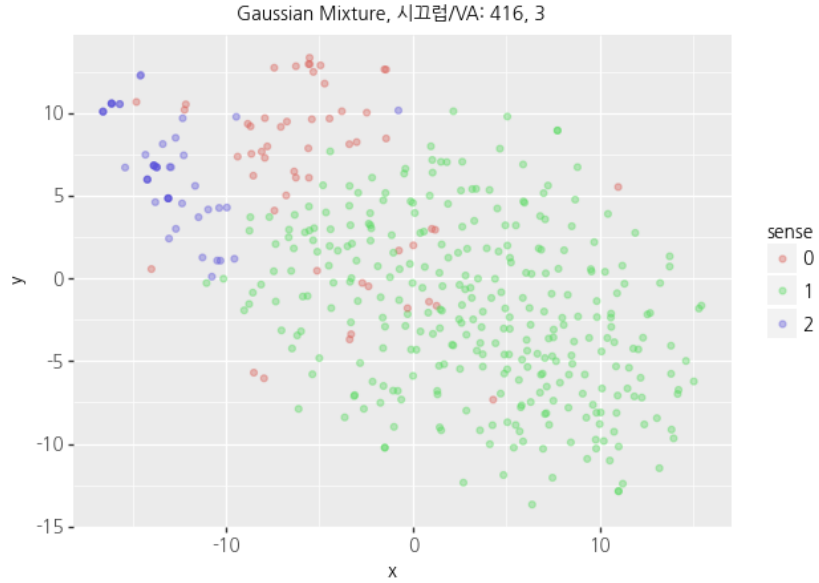

Bayesian Gaussian Mixture, 쓰러지/VV: 1012, 3
sense
0
1
2
x
y

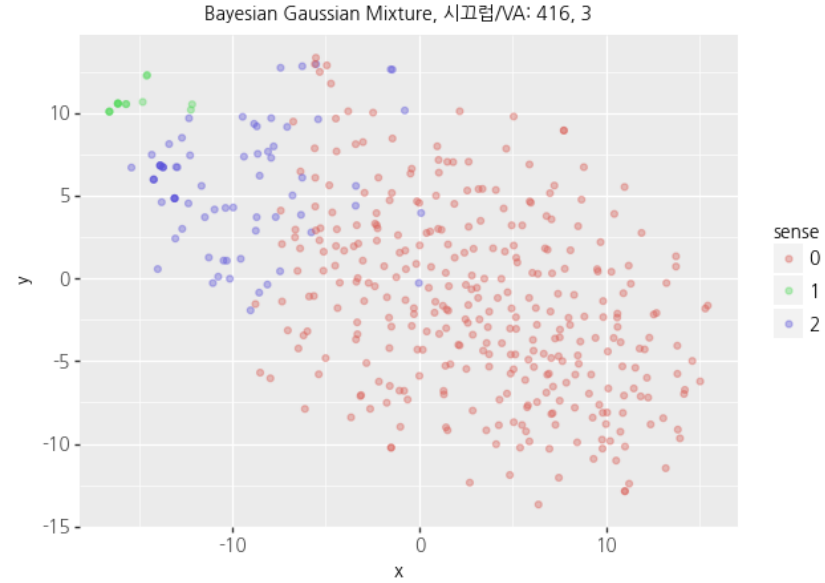

K-Means, 과감하/VA: 408, 2
sense
0
1
x
y

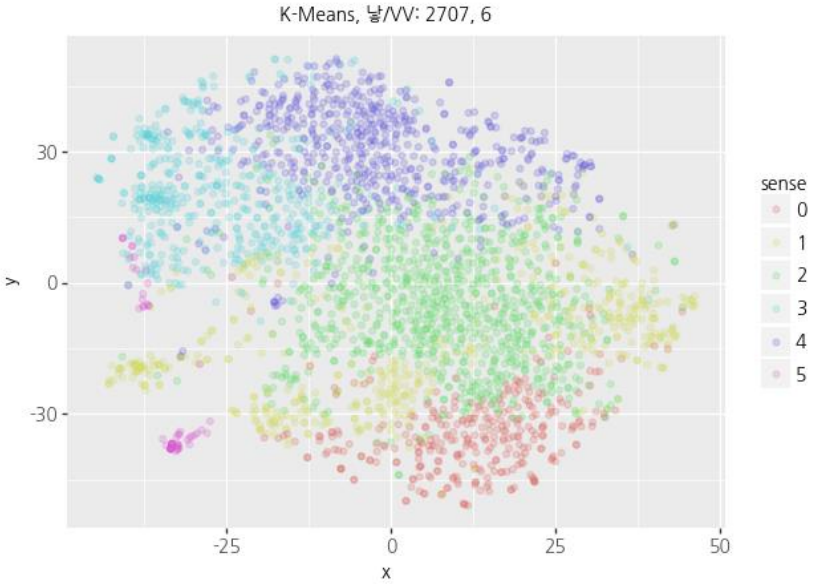

Gaussian Mixture, 과감하/VA: 408, 2
sense
0
1
x
y

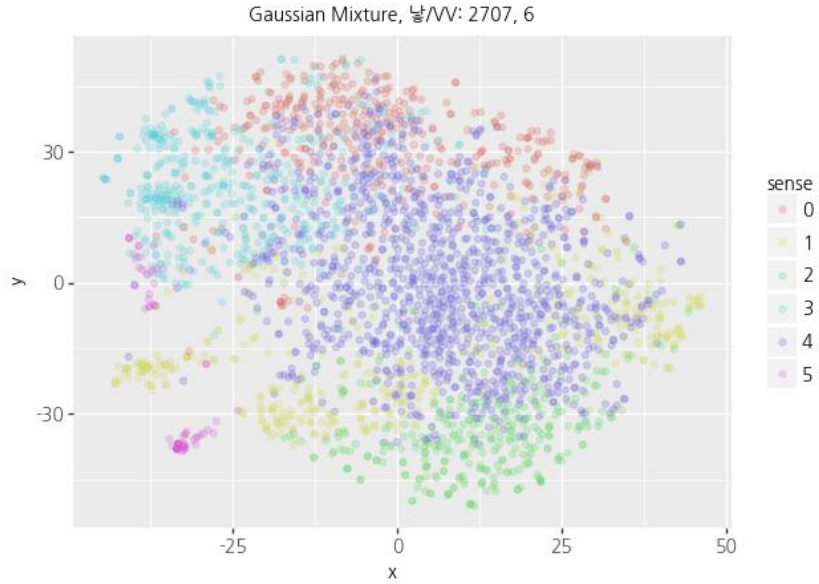

Bayesian Gaussian Mixture, 과감하/VA: 408, 2
sense
0
1
x
y

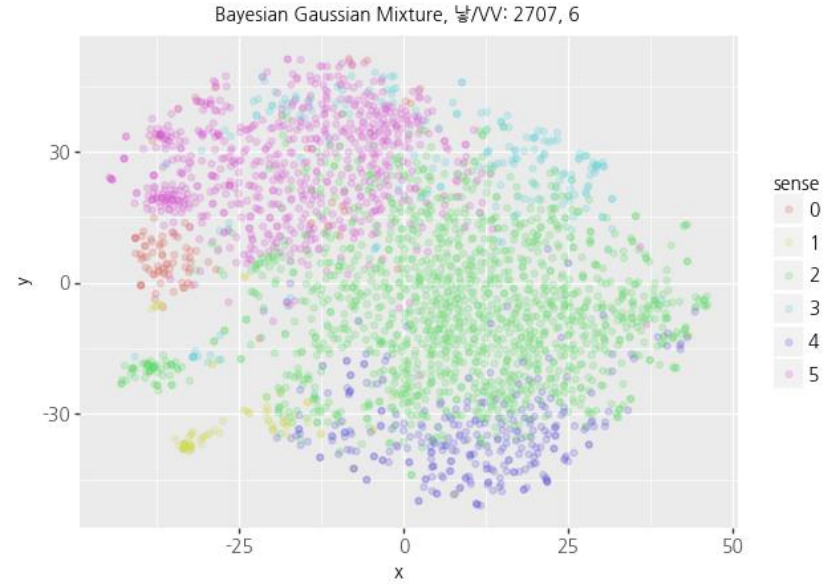

K-Means, 시끄럽/VA: 416, 3
sense
0
1
2
x
y

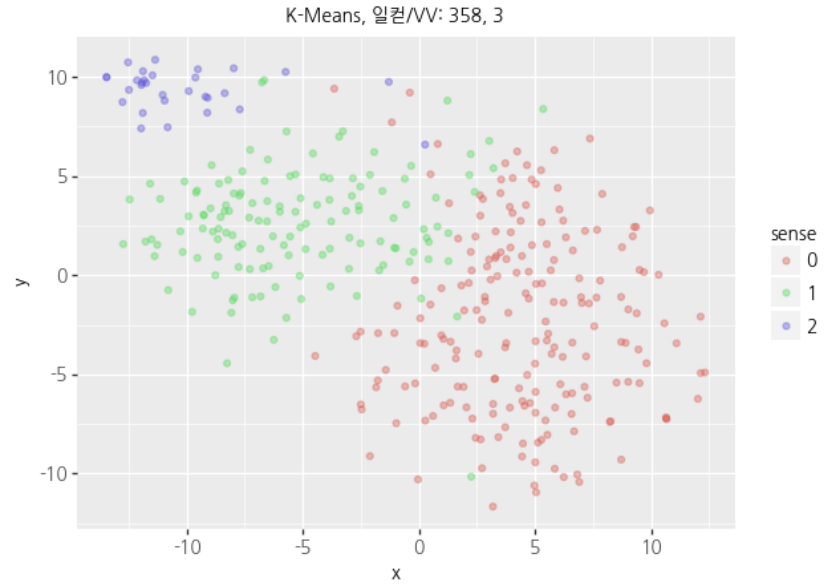

Gaussian Mixture, 시끄럽/VA: 416, 3
sense
0
1
2
x
y

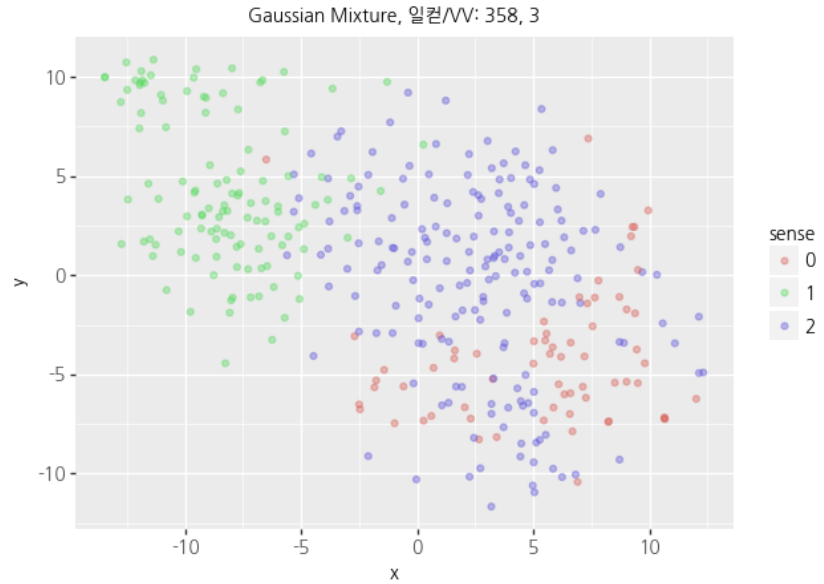

Bayesian Gaussian Mixture, 시끄럽/VA: 416, 3
sense
0
1
2
x
y

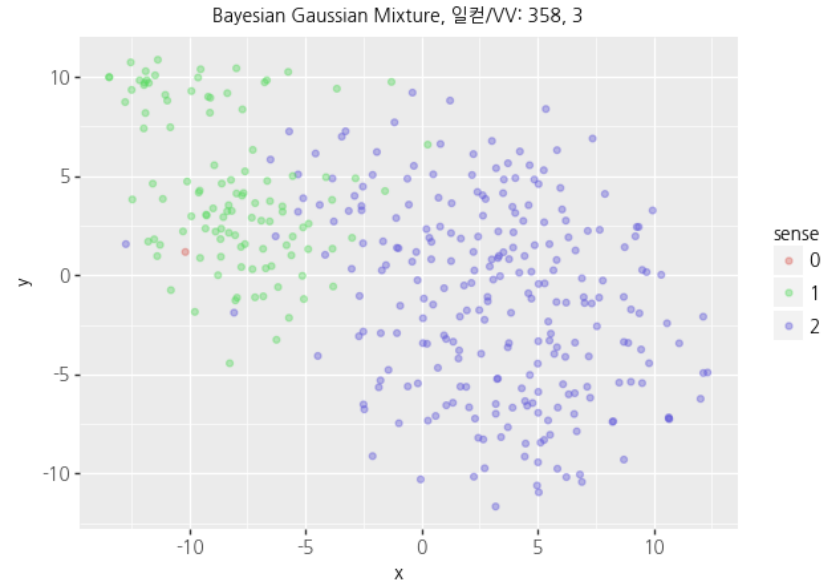

K-Means, 낮/VV: 2707, 6
sense
0
1
2
3
4
5
x
y

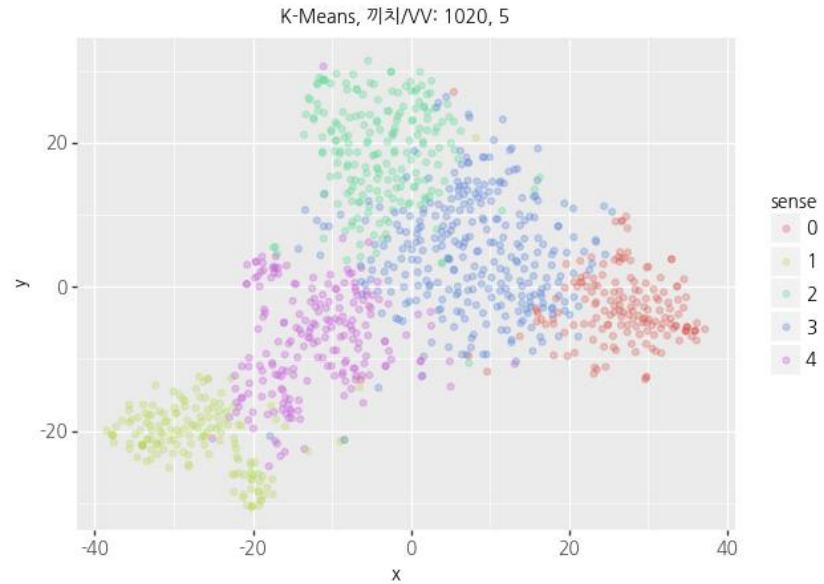

Gaussian Mixture, 낮/VV: 2707, 6
sense
0
1
2
3
4
5
x
y

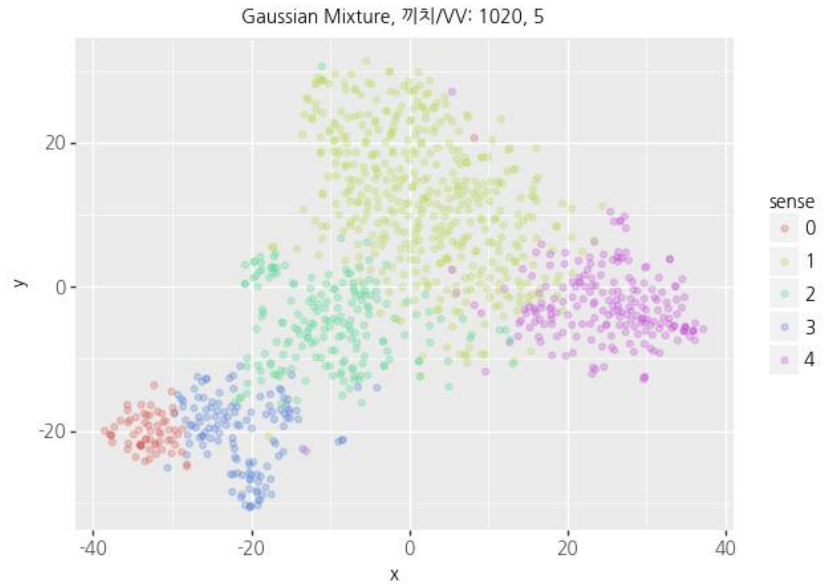

Bayesian Gaussian Mixture, 낮/VV: 2707, 6
sense
0
1
2
3
4
5
x
y

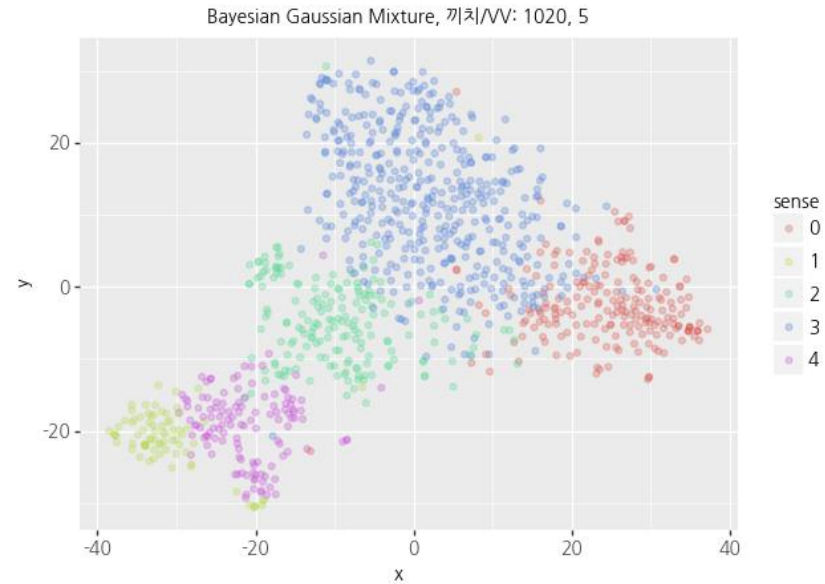

K-Means, 일컫/VV: 358, 3
sense
0
1
2
x
y

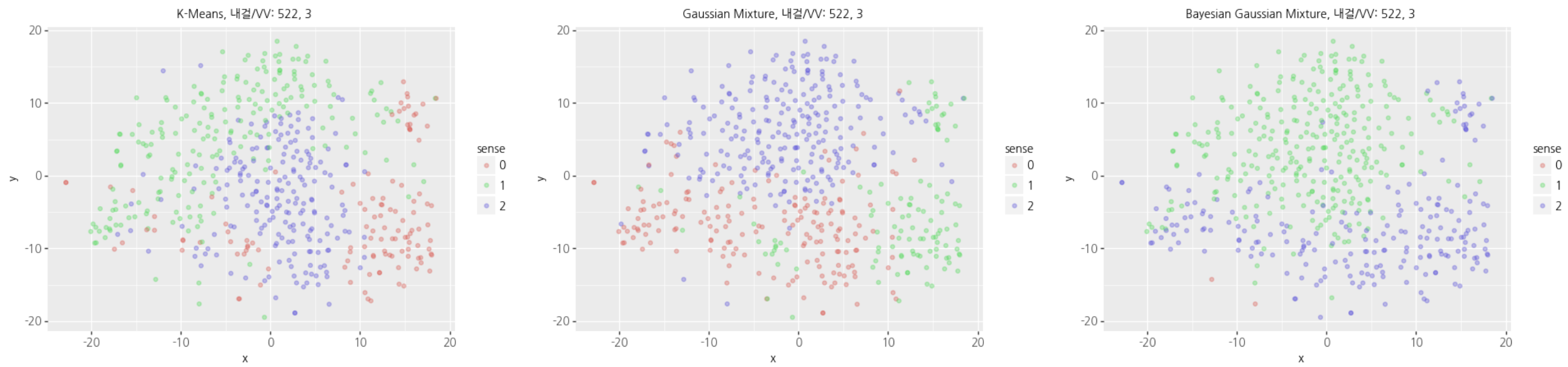

Gaussian Mixture, 일컫/VV: 358, 3
sense
0
1
2
x
y


Bayesian Gaussian Mixture, 일컫/VV: 358, 3
sense
0
1
2
x
y


K-Means, 끼치/VV: 1020, 5
sense
0
1
2
3
4
x
y


Gaussian Mixture, 끼치/VV: 1020, 5
sense
0
1
2
3
4
x
y


Bayesian Gaussian Mixture, 끼치/VV: 1020, 5
sense
0
1
2
3
4
x
y



- '내걸/VV'의 경우 BGM에서 부익부빈익빈 현상이 나타남.
- 그 외의 측면에서도 K-Means, GMM, BGM 셋 다 상당히 다른 결과를 내놓았다는 것이 좀 특이.
- 군집화 결과에 대한 종합
  - ① 3가지 방법이 상당히 유사한 결과를 낳은 경우가 꽤 많음.
  - ② K-Means와 GMM에 비해 BGM은 군집의 크기가 상당히 편향적으로 나타나는 경향이 있고, 그것이 실제에 부합되는 일이 많은 듯.
  - ③ (부인부빈인빈 문제를 차치하면) GMM과 BGM의 결과가 꽤 유사하고 K-Means는 이와 사뭇 다른 경우가 많음.

군집화 정확도 평가: '일컫/VV'의 경우

- K-Means가 부여한 군집 번호 통계: k0=194, k1=134, k2=30
  - K-Means 군집 번호 대 우리말샘 의항 번호 대응: k0=gt1, k1=gt2, k2=gt3
  - 정확도: 198/358 = 0.55
- GMM이 부여한 군집 번호 통계: g0=66, g1=113, g2=179
  - GMM 군집 번호 대 우리말샘 의항 번호 대응: g0=gt3, g1=gt2, g2=gt1
  - 정확도: 188/358 = 0.53
- BGM이 부여한 군집 번호 통계: b0=1, b1=110, b2=247
  - BGM 군집 번호 대 우리말샘 의항 번호 대응: b0=gt3, b1=gt2, b2=gt1
  - 정확도: 234/358 = 0.65
- GMM과 BGM은 군집 번호와 의항 번호의 대응 양상이 동일하고
- 이 둘은 K-Means의 대응 양상과는 다름.
- 세 군집화 방법 가운데 BGM이 가장 높은 정확도를 보임: 65%

혼동행렬을 통한 군집화 오류 분석

K-M	k0	k1	k2	계
gt1	136	70	4	210
gt2	56	62	26	144
gt3	2	2	0	4
계	194	134	30	358

G	g2	g1	g0	계
gt1	123	47	40	210
gt2	53	65	26	144
gt3	3	1	0	4
계	179	113	66	358

B	b2	b1	b0	계
gt1	167	42	1	210
gt2	77	67	0	144
gt3	3	1	0	4
계	247	110	1	358

- 각 표의 9개 셀 가운데, 적색으로 표시된 대각선 셀들은 군집화 알고리즘이 옳게 판단한 경우.
- K-Means는, gt1인데 다른 것으로 오판한 것(74개)이 다른 것을 gt1으로 오판한 것(58개)보다 많음: gt1의 크기를 과소평가(210: 194)
- GMM도, gt1인데 다른 것으로 오판한 것(87개)이 다른 것을 gt1으로 오판한 것(56개)보다 많음: gt1의 크기를 과소평가(210: 174)
- BGM은, gt1인데 다른 것으로 오판한 것(43개)이 다른 것을 gt1으로 오판한 것(80개)보다 적음: gt1의 크기를 과대평가(210: 247)

군집 통합을 통한 정확도 계산

- 군집화 알고리즘이 별개의 군집으로 간주한 것을 하나의 군집으로 합쳐서 우리말샘과의 대응을 생각해 볼 수도 있음.
  - '일컫-'의 우리말샘 의항3은 말뭉치에 출현 빈도가 매우 낮으므로
  - 군집화 알고리즘이 이에 해당한다고 판단한 것들을 다른 군집에 합쳐 버리는 것.
  - 즉 K-Means의 경우 k0=gt1, k1=gt2, k2=gt2로 대응시키는 것.
- 이렇게 할 경우 혼동 행렬은 아래와 같이 되고
- 정확도는 K-Means: 0.66, GMM: 0.64, BGM: 0.65가 됨.

K-M	k0	k1	k2	계
gt1	136	70	4	210
gt2	56	62	26	144
gt3	2	2	0	4
계	194	134	30	358

G	g2	g1	g0	계
gt1	123	47	40	210
gt2	53	65	26	144
gt3	3	1	0	4
계	179	113	66	358

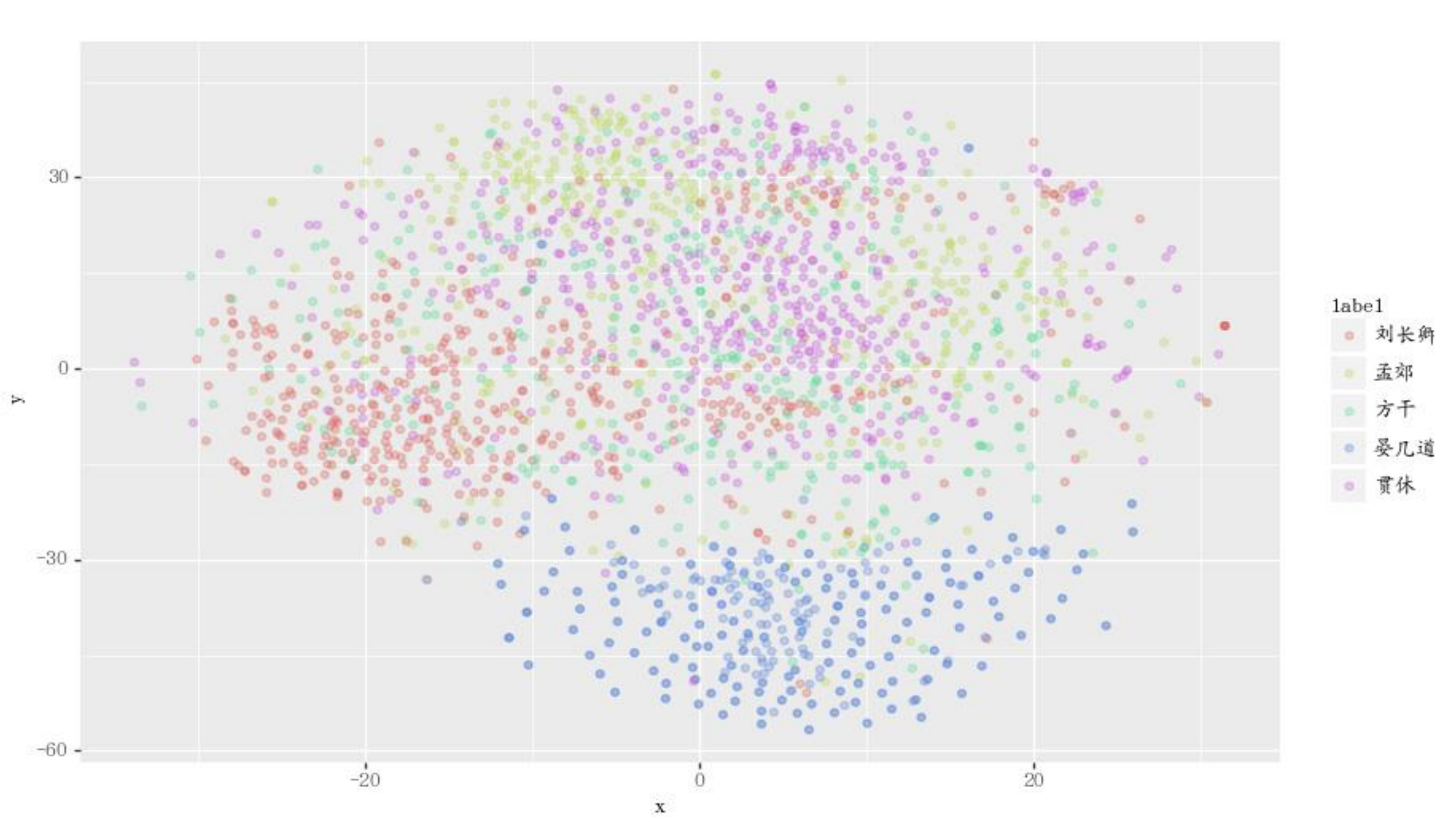
B	b2	b1	b0	계
gt1	167	42	1	210
gt2	77	67	0	144
gt3	3	1	0	4
계	247	110	1	358

4. 한시 분류
자료 수집

- 诗词名句網 홈페이지(http://www.shicimingju.com/)에서 13,086명 시인의 291,934수의 한시를 수집.
  - 작품이 100수 이상인 시인은 556명, 200수 이상인 시인은 297명,
  - 300수 이상인 시인은 208명, 400수 이상인 시인은 143명,
  - 500수 이상인 시인은 106명, 1000수 이상인 시인은 36명.
  - 이 가운데 간혹 중국 이외의 시인이 포함되어 있으나 대다수는 중국 시인.
- 한국 한시는『동문선』권4~10에 수록된 시 1892수를 수집.
- 일본의 懷風藻, 本朝文粹, 文華秀麗集, 本朝麗藻, 和漢朗詠集, 本朝一人一首, 小倉百人一首, 本朝無題詩, 凌雲集, 菅家文草, 經國集, 田氏家集, 江吏部集, 法性寺關白御集, 雜言奉和 등의 한시집에서 총 3572수를 수집.

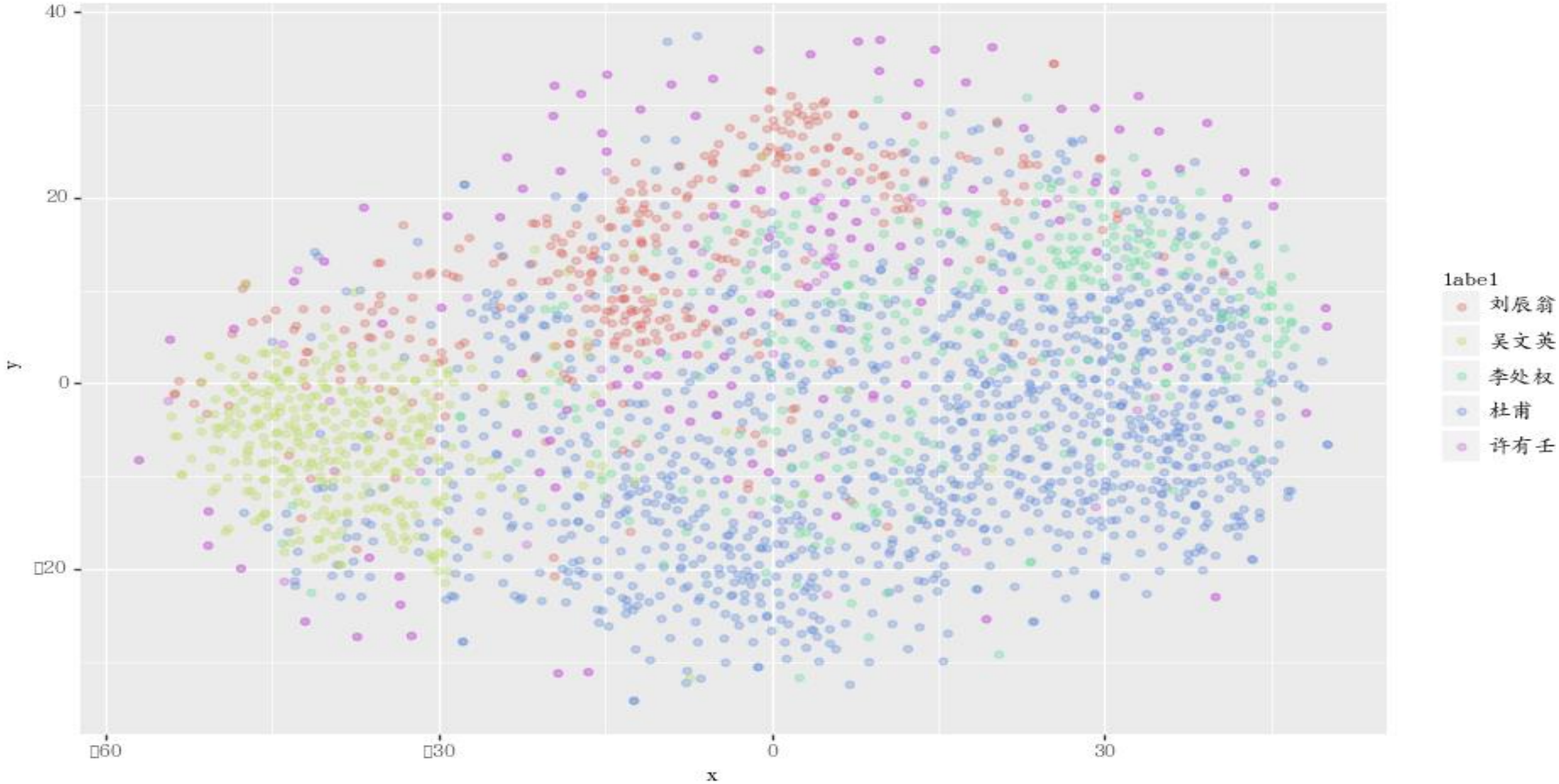
자료 전처리 및 벡터화

- 수집된 자료에서 중국 한시는 간체자, 한국 한시는 번체자, 일본 한시는 일본식 한자체로 되어 있어서, 이를 통일할 필요가 있음.
  - 간체자와 번체자 사이에 1대다 대응 관계가 간혹 있으므로(예컨대 云:云/雲, 发:發/髮 등) 간체자를 번체자로 자동 변환할 수는 없음.
  - 그래서 한국 한시와 일본 한시도 간체자로 변환하여 처리하였음.
- 수집된 한중일의 한시 전체를 Word2Vec 알고리즘에 넣어서 신경망 훈련 과정을 거쳐서 각 글자(한자)의 벡터 표상을 얻었음.
  - 약 30만 수의 한시에 포함된 한자의 유형은 모두 12,501개.
- 그 다음에 이 글자 벡터를 바탕으로 하여, Fast Sentence Embedding (FSE)라는 알고리즘을 이용하여 각 한시의 벡터 표상을 얻었음.
  - FSE 알고리즘은 Word2Vec에 의해 얻어진 단어/글자 벡터를 바탕으로 하여 이를 종합하여 이 단어/글자들의 연쇄인 문장의 벡터를 산출.
  - 의미가 같거나 비슷한 단어/글자들이 비슷한 순서로 많이 나타날수록 문장 벡터도 유사하게끔 되어 있음.


30
0
-30
-60
y
-20
0
20
x
label
刘长卿
孟郊
方干
晏几道
贯休

贯休, 晏几道, 刘长卿, 方干, 孟郊

- 이 5명 시인의 작품 2130수를 2차원 평면에 그래프로 나타내면 앞 슬라이드와 같이 됨.
- 晏几道의 작품들(청색)은 하단에 몰려 있고, 다른 작가의 작품들과 거의 섞이지 않았음.
- 그에 비하면 나머지 4명의 작품들은 좀 섞여 있기는 하지만
- 그래도 刘长卿(적색)은 좌측, 孟郊(연두)는 상단, 贯休(보라)는 그 우측에 몰려 있음.
- 어떤 작품이 다른 작품과 얼마나 유사한지도 이런 그래프를 통해 알아볼 수 있음.
  - 하단에는 대부분 晏几道(청색)의 작품들이 몰려 있지만
  - 그 안에 刘长卿(적색)의 작품이 몇 개 포함되어 있음.
  - 이 몇 개의 작품은 刘长卿의 다른 작품들과 달리 晏几道의 작품과 유사도가 높은 작품이라고 해석할 수 있음.

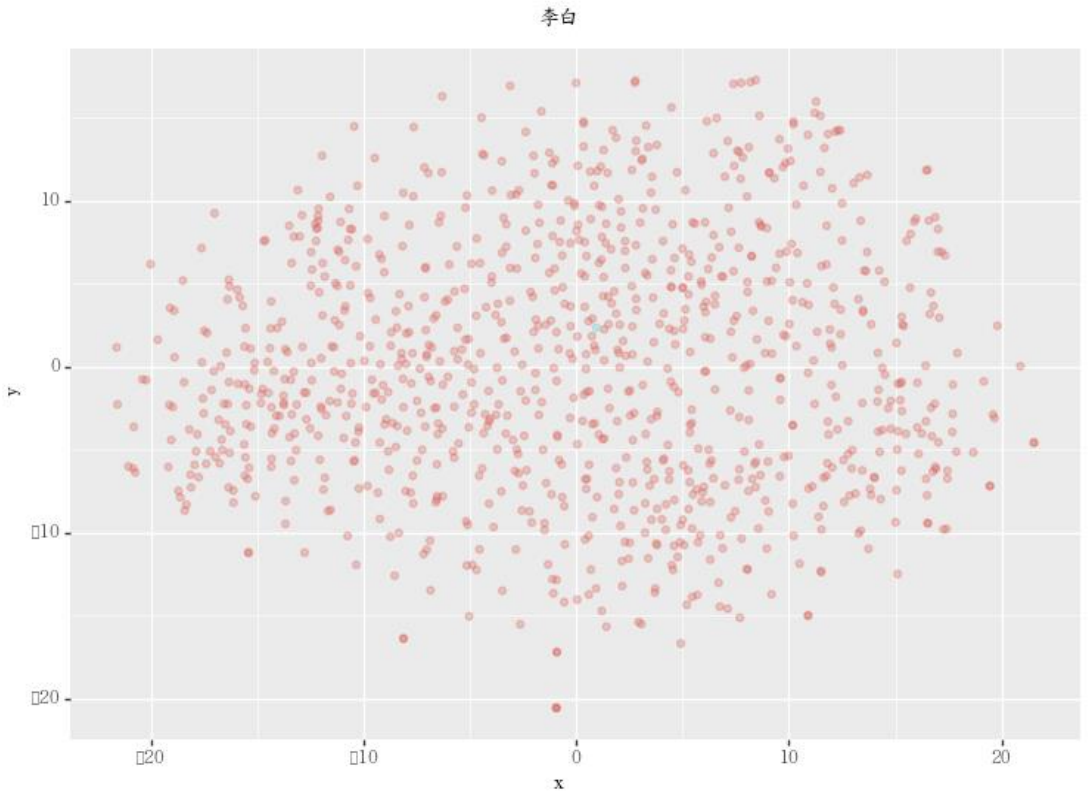

40
20
0
□20
y
□60
□30
0
30
x
label
刘辰翁
吴文英
李处权
杜甫
许有壬

吴文英, 刘辰翁, 李处权, 许有壬, 杜甫

- 이 5명의 작품 2499수를 그래프화한 것: 앞 슬라이드
- 吴文英(연두)의 작품들은 좌측에 몰려 있고
- 许有壬(보라)의 작품들은 맨 위 테두리에 있으며
- 刘辰翁(적색)의 작품들은 그 테두리 안쪽의 상단에 몰려 있음.
- 李处权(초록)과 杜甫(청색)의 작품들은 그에 비하면 넓게 퍼져 있기는 하지만
- 대체로 杜甫의 작품들은 右下에 쏠려 있고
- 李处权의 작품들은 그보다 안쪽에 분포하고 있음.

작가별 중심점과 이로부터의 거리

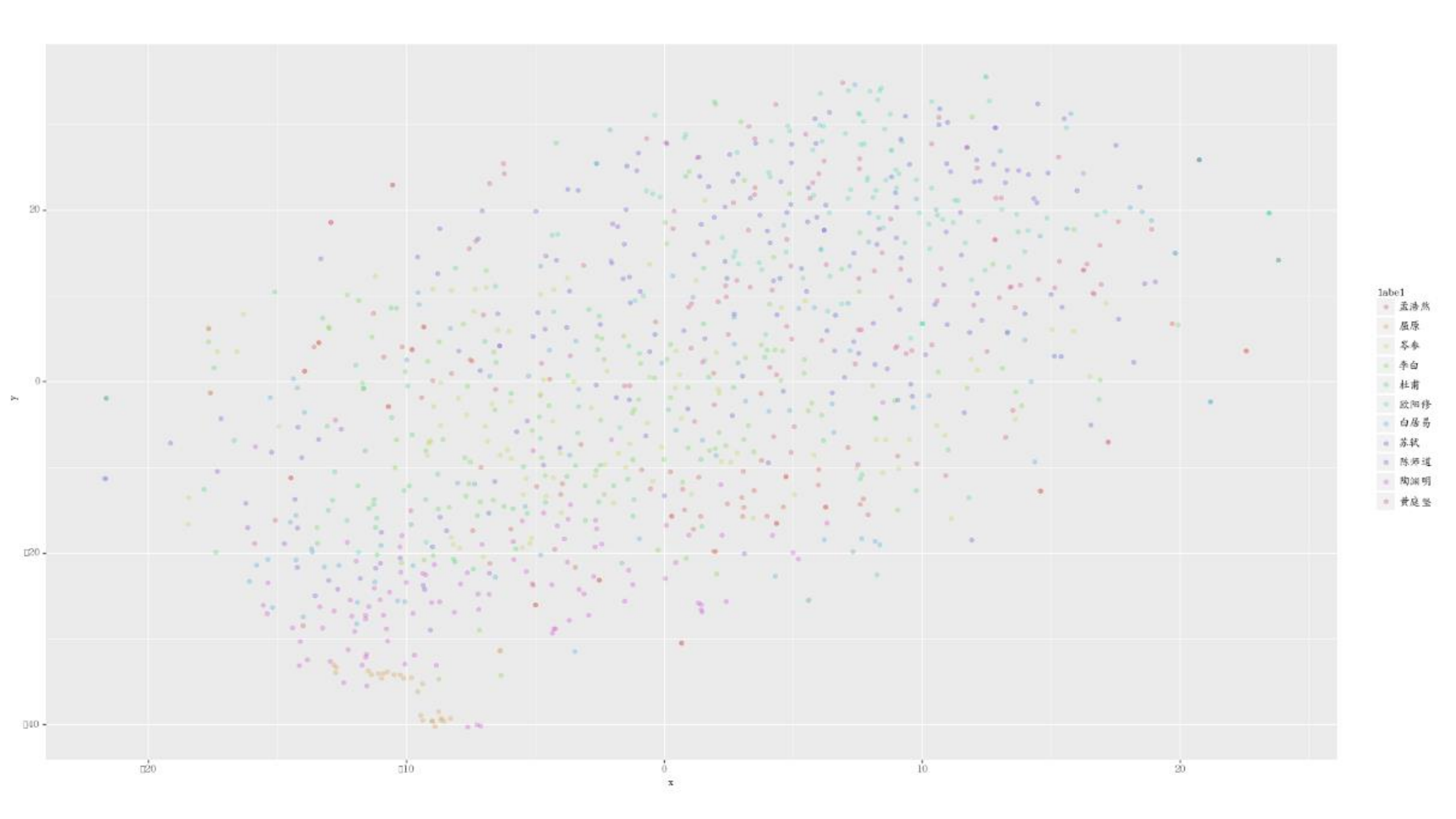
- 이렇게 2차원 평면상에 그래프로 그려 보면, 각 작가의 작품들이 얼마나 동질적인지 이질적인지도 알아볼 수 있음.
- 诗词名句網에 李白의 작품은 981수.
- 이 작품들을 2차원 평면상에 배열하고, 모든 작품들의 중심점을 찍어 보면 우측과 같음.
- 적색 점들이 李白의 실제 작품이고
- 중앙의 초록색 점이 중심점.
- 각 작가의 작품들의 중심점으로부터의 거리를 구하고 그 평균을 구했을 때
- 이 평균값이 크면 이 작가의 작품들은 이질적이라고 할 수 있고
- 이 평균값이 작으면 이 작가의 작품들은 동질적이라고 할 수 있



순위	시인	작품 수	거리 평균	순위	시인	작품 수	거리 평균
1	陈人杰	30	0.5042	1382	先秦无名	114	1.2728
2	张伯淳	22	0.5586	1383	石延年	89	1.2769
3	冯取洽	25	0.5685	1384	曹植	140	1.2839
4	李道纯	60	0.5702	1385	宋度宗	22	1.2900
5	冯尊师	24	0.5719	1386	李邴	42	1.2972
6	五迈	76	0.5726	1387	齐唐	28	1.3067
7	赵以夫	68	0.5802	1388	杨徽之	24	1.3178
8	刘处玄	63	0.5945	1389	萧德藻	26	1.3341
9	凌云翰	43	0.6029	1390	程敦厚	26	1.3404
10	京镗	45	0.6043	1391	欧阳鈇	28	1.3410
11	张之翰	67	0.6084	1392	谌祜	119	1.3603
12	李孝光	24	0.6180	1393	应璩	36	1.3704
13	葛郯	29	0.6302	1394	***	383	1.4197
14	吴文英	342	0.6351	1395	吴光	25	1.4424
15	朱唏颜	37	0.6379	1396	释惠崇	120	1.4589
16	刘将孙	21	0.6403	1397	王予可	22	1.4633
17	魏初	44	0.6411	1398	王随	64	1.4689
18	黄裳	57	0.6419	1399	李缜	33	1.9363
19	吴存	30	0.6483	1400	张釜	70	1.9608
20	王 *	31	0.6493	1401	释晓莹	126	2.2027

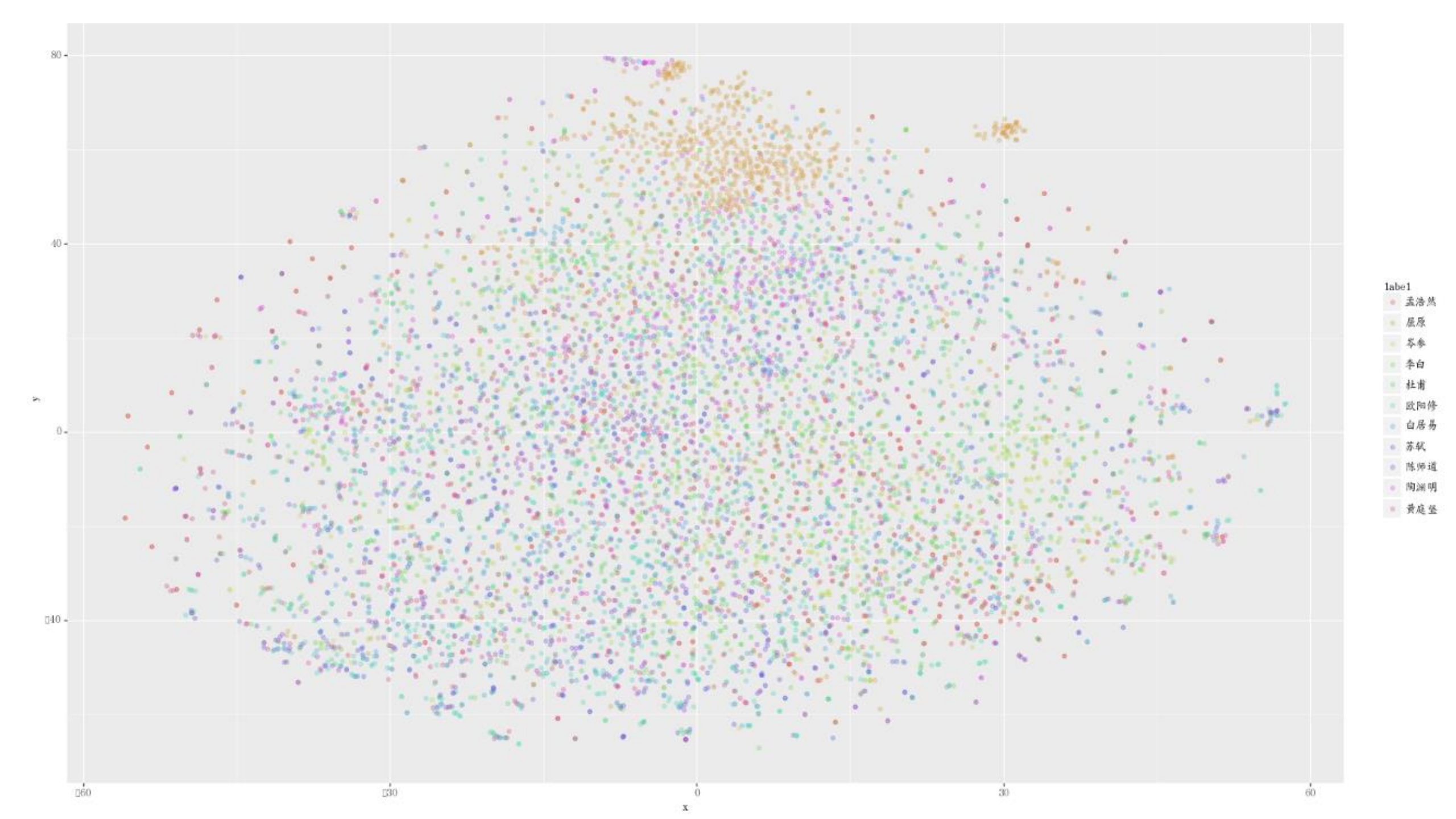
작가별 작품 산포도와 작품 수

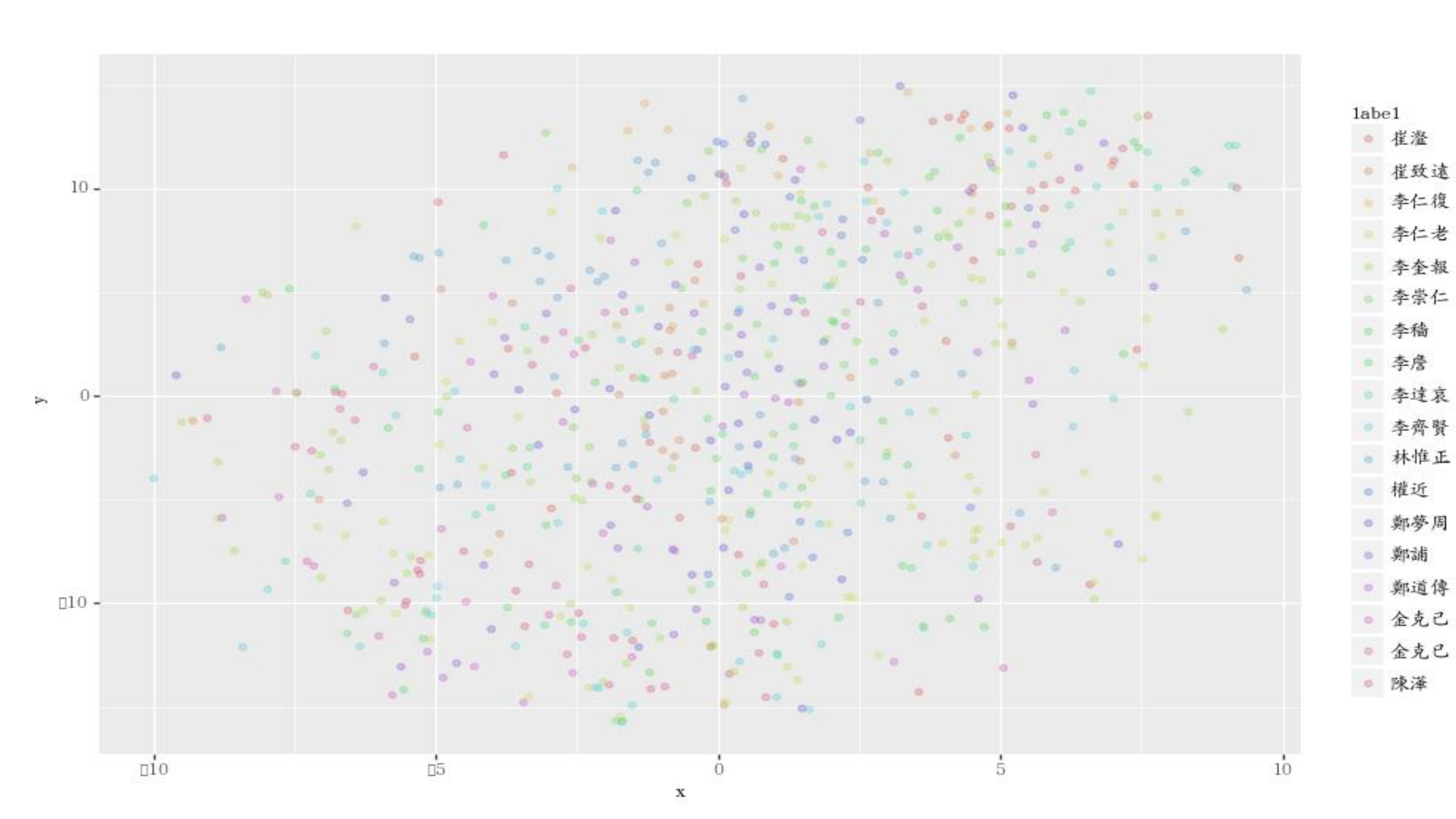
- 이 거리 평균값은 작품 수와 함께 고려해야 함.
- 작품 수가 적을 때는 작품들이 비교적 동질적이기가 쉬우나
- 작품 수가 많아질수록 동질성을 유지하기가 쉽지 않게 됨.
- 따라서 작품 수가 비교적 적은 작가는 거리 평균값이 작은 것이 당연.
- 작품 수가 많은데도 거리 평균값이 작은 작가는 매우 뚜렷하게 작품의 동질성을 유지하고 있는 것.
  - 14위 吴文英(342수), 22위 张炎(300수), 72위 张翥(200수), 74위 刘辰翁(347수), 80위 高适(205수), 83위 李处权(321수), 104위 王恽(248수), 112위 许有壬(301수), 136위 杜甫(1188수), 144위 贯休(517수), 150위 晏几道(401수) 등이 그러한 경우에 속함.
- 거리 평균값이 높아서 작품들이 이질적인 편에 속하는 시인
  - 1384위 曹植(140수), 1370위 张抡(109수), 1366위 张尧同(100수), 1365위 高似孙(191수), 1362위 宋高宗(146수), 1360위 曹丕(54수), 1348위 萧衍(105수), 1343위 陶弼(220수), 1335위 杨备(116수), 1328위 沈约(269수), 1326위 贺知章(26수), 1321위 王志道(31수) 등.
  - 작가가 “***”, “先秦无名”, “无名氏”, “X” 등으로 표시된 것은 사실은 1명의 작가가 아니므로 거리 평균값이 큰 것이 당연.


20
0
−20
−40
y
−20
−10
0
10
20
x
label
孟浩然
屈原
岑参
李白
杜甫
欧阳修
白居易
苏轼
陈师道
陶渊明
黄庭坚

李白, 苏轼, 杜甫, 白居易, 陶渊明, 屈原, 孟浩然, 岑参, 欧阳修, 黄庭坚, 陈师道

- 앞 슬라이드는 이 11명의 작품들을 작가당 100수씩 뽑아 하나의 그래프상에 나타내 본 것.
- 屈原의 작품들이 左下 구석에 집중되어 있음이 특징적.
- 屈原의 작품 수가 적은 탓도 있겠으나(27수)
- 평균 거리값도 0.6914(46위)로 역시 매우 낮은 편.
- 즉 屈原은 다른 작가들에 비해 작품들이 동질적.
- 다음 슬라이드는 동일한 11명 시인의 (작품 전체가 아니라) 詩句 8221개를 가지고 그래프를 그려 본 것.
- 역시 屈原의 詩句들이 상단 중앙에 몰려 있음


80
40
0
−40
y
−60
−30
0
30
60
x
label
孟浩然
屈原
岑参
李白
杜甫
欧阳修
白居易
苏轼
陈师道
陶渊明
黄庭坚


10
0
□10
y
□10
□5
0
5
10
x
label
崔澄
崔致遠
李仁復
李仁老
李奎報
李崇仁
李穡
李詹
李達哀
李齊賢
林惟正
權近
鄭夢周
鄭誧
鄭道傳
金克己
金克已
陳澕

동문선

- 앞 슬라이드는『동문선』에서 한시 작품이 20수 이상인 시인들의 작품을 그래프화한 것.
- 한 작가의 작품들이 한 군데에 몰려 있지 않고, 여러 작가의 작품들이 이러저리 뒤섞여 있는 모습을 볼 수 있음.
- 『동문선』에서 한시 작품이 10수 이상인 시인들을 중심값으로부터의 거리 평균에 따라 배열하면 다음과 같음.

순위	시인	작품 수	거리 평균	순위	시인	작품 수	거리 평균
1	釋天因	18	0.6332	23	鄭夢周	33	0.8747
2	尹紹宗	16	0.7045	24	崔瀣	33	0.8784
3	釋月窓	19	0.7193	25	權漢功	15	0.8787
4	李穀	11	0.7283	26	李存吾	11	0.8817
5	李藏用	11	0.7488	27	柳淑	18	0.8852
6	崔滋	11	0.7723	28	卞仲良	11	0.9035
7	李仁復	21	0.7782	29	鄭樞	16	0.9154
8	洪侃	15	0.7811	30	鄭知常	13	0.9249
9	林椿	19	0.8116	31	李穡	55	0.9331
10	權近	20	0.8198	32	崔惟淸	14	0.9405
11	金富軾	18	0.8225	33	陳澕	42	0.9427
12	趙浚	11	0.8300	34	郭預	12	0.9553
13	韓脩	12	0.8348	35	李齊賢	64	0.9617
14	李崇仁	35	0.8514	36	李奎報	65	0.9628
15	卞季良	13	0.8531	37	崔致遠	29	0.9690
16	李達哀	20	0.8545	38	金克己	52	0.9720
17	安軸	12	0.8563	39	釋圓鑑	17	0.9816
18	鄭道傳	22	0.8666	40	白元恒	17	0.9849
19	偰遜	19	0.8666	41	李仁老	80	1.0296
20	李詹	27	0.8701	42	姜碩德	13	1.0580
21	鄭誧	37	0.8739	43	金富軾	15	1.0884
22	林惟正	46	0.8745				

맺는 말

- 본 발표에서는 종래의 표면적인 문장 유사도 지표(편집거리)의 한계를 지적하고
- Word2Vec 알고리즘을 통해 단어/글자를 벡터화하고
- 이 벡터를 바탕으로 한 심층적인 의미상의 유사도 지표를 측정하는 방법을 제시하였음.
- 그리고 FSE 알고리즘을 이용하여 단어/글자뿐 아니라 그 연쇄인 문장/한시도 벡터화하는 방법을 제시하였음.
- 분포의미론의 전제 위에서, 언어요소의 주위 문맥이 벡터화의 단서가 됨.
- 이 문맥/분포를 나이브하게 문장 전체 또는 타깃 요소와 선조적으로 인접한 요소로 생각하기보다, 타깃 요소와 통사적으로 긴밀한 관계를 맺는 요소로 생각하면 더 좋은 결과를 얻을 수 있음.