Sub Word Level Embedding이 Lstm Classifier의 성능 향상에 도움을 주는가?

- 2018년 05월 09일
- 언어학과
- 이 한 결

Contents

Overview(Goal)

- Create various levels of word embeddings
  - word2vec – skipgram/cbow **(Google)**
  - Glove **(Standford NLP Group)**
  - fastText N-gram based embeddings (**Facebook**)
- Evaluate its performance:
  - Model : **-> LSTM classifier**
  - Task :**-> Sentiment Analysis**

Why : (1) Sub word Level?

- 임베딩의 단위 : 고립어(영어) vs 교착어(한국어)

	"He is a student"	"그는 학생이다."
Character	H / eVi / sVaVs / t / u / d / e / n / t	ㄱ / ㅡ / ㄴ / ㅡ / ㄴV ㅎ / ㅏ / ㄱ / ㅅ / ㅐ / ㅇ / ㅇ / ㅣ / ㄷ / ㅏ
Syllable	HeVisVaVstu / dent [hi] [Iz] [ə] [stu:] / [dnt]	그 / 는V학 / 생 / 이 / 다 [geu] / [neun] [hak] / [saeng] / [i] / [da]
Word	HeVisVaVstudent	그는V학생이다
Morpheme	HeVisVaVstudent Pronoun Verb Indefinite article Noun	그 / 는V학생 / 이다 Pronoun / Postposition Noun / Verb

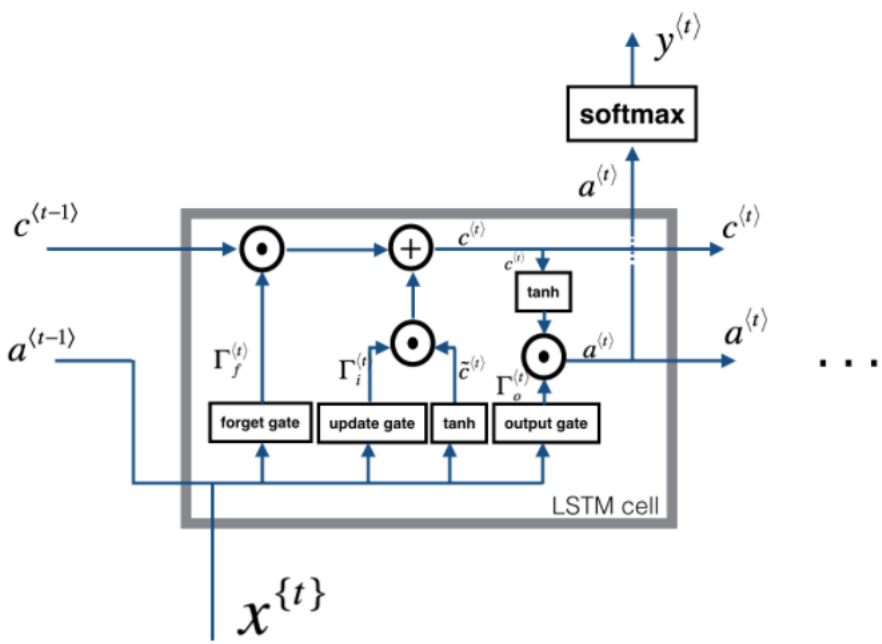
- 단어 단위의 임베딩은 한국어에 적합하지 못함.
  - 임베딩 학습을 위한 각 단어당 최소 출현빈도가 보장되기 힘듬
  - 단순히 어미가 변화된 단어일지라도 다른 공간에 맵핑될수 있음

Why ? (2) Sentiment Analysis

- 기존의 임베딩 질을 평가하는 방법의 한계
  - 기존의 논문들은 대부분 semantic similarity/ analogy task 에서만 임베딩의 질을 측정했으나 , 어떤 데이터 셋을 쓰는지에 따라, 성능 평가가 달라짐
    - ex) Google Analogy vs WSW353
  - 단어 임베딩은 end-end network의 embedding layer에서 가장 많이 사용됨
    - ex) Name entity recognition/ machine translation/ part of speech tagging 등
    - 대부분의 딥러닝 기반의 nlp system에서 embedding layer를 사용함.
  - End-End System에서의 성능을 측정하는 것이 더 타당
    - Sentiment Analysis는 비교적 점수와 질과의 상관관계가 명확함 (c.f. Bleu Score)
    - Labeled Data Set을 구하기 쉬움

Why ? (3) LSTM Classifier

- What Classifier? LSTM
  - Sequence model : RNN / GRU / LSTM / BRNN 가운데 **LSTM**을 택
  - Vanilla RNN 을 사용하지 않는 이유 ?
    - Sequence length가 길어지면 임베딩의 질과 관련없이 vanishing/ exploding gradient 문제에 노출되기 때문



$$\Gamma_f^{\langle t\rangle} = \sigma(W_f[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_f)$$

$$\Gamma_u^{\langle t\rangle} = \sigma(W_u[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_u)$$

$$\tilde{c}^{\langle t\rangle} = \tanh(W_C[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_C)$$

$$c^{\langle t\rangle} = \Gamma_f^{\langle t\rangle} \circ c^{\langle t-1\rangle} + \Gamma_u^{\langle t\rangle} \circ \tilde{c}^{\langle t\rangle}$$

$$\Gamma_o^{\langle t\rangle} = \sigma(W_o[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_o)$$

$$a^{\langle t\rangle} = \Gamma_o^{\langle t\rangle} \circ \tanh(c^{\langle t\rangle})$$

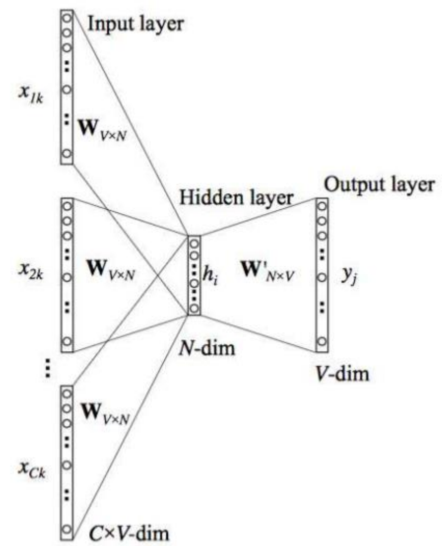
Embedding Model

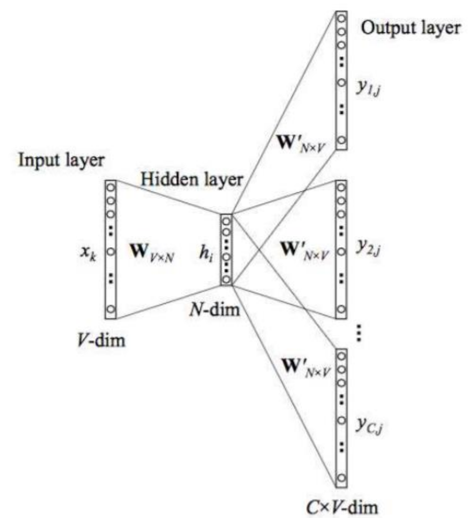
- Word2Vec (objective)
  - CBOW(count bags of words)

$$\frac{1}{V}\sum_{t=1}^{V}\log p(m_t|m_{t-\frac{c}{2}}...m_{t+\frac{c}{2}})$$

  - SkipGram

$$\frac{1}{V}\sum_{t=1}^{V}\sum_{j=t-c,j\neq t}^{t+c}\log p(m_j|m_t)$$

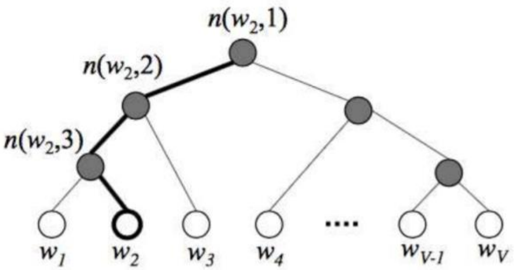




Embedding Model

- Word2Vec(loss function)
  - Hierarchical softmax

    (use Huffman tree)



  - **Negative Sampling**

$$E = -\log \sigma(\mathbf{v}'_{w_O}{}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-\mathbf{v}'_{w_j}{}^T \mathbf{h})$$

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_j)^{3/4} \right)}$$

Embedding Model

- Glove
  - Ratio $F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$ $F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$
  - Symmetry $w -> \tilde{w}$ and $X -> X^T$ $w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$
  - Loss Function

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2$$

Embedding Model

- SGNS ~ Glove
  - SGNS can be viewed as shifted PMI

$$W_i \cdot C_j^T = PMI(w_i, c_j) - \log k$$
$$= \log \#(w_i, c_j) - \log \#(w_i) - \log \#(c_j) + \log \Sigma_w \#(w) - \log k.$$

  - Glove : bias term b_{W_i} and b_{C_j} may converge to the values in SGNS $\log \#(w_i)$ and $\log \#(c_j)$

$$l_G(w_i, c_j) = f(\#(w_i, c_j)) \left(W_i \cdot C_j^T + b_{W_i} + b_{C_j} - \log \#(w_i, c_j)\right)^2$$

$$W_i \cdot C_j^T = \log \#(w_i, c_j) - b_{W_i} - b_{C_j}$$

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Embedding Model

- FastText
  - equation $$\sum_{t=1}^{T} \sum_{c \in C_t} logp(w_c|w_t)$$

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}}$$

$$s(w,c) = \sum_{g \in g_w} z_g^T v_c$$

$$g_{소프트웨어} = \{<소프, 소프트, 프트웨,$$

$$트웨어, 웨어>, <소프트웨어>\}$$

Experiment Design

① create embeddings

- Word2Vec / Glove comparison

  단어단위 vs 형태소 단위 비교

- FastText n-gram comparison

  Syllable 단위 vs 자/모단위

	skip-gram	cbow	glove	fastText	
				syllable	jamo
word	O	O	O	O	O
morpheme	O	O	O	O	△

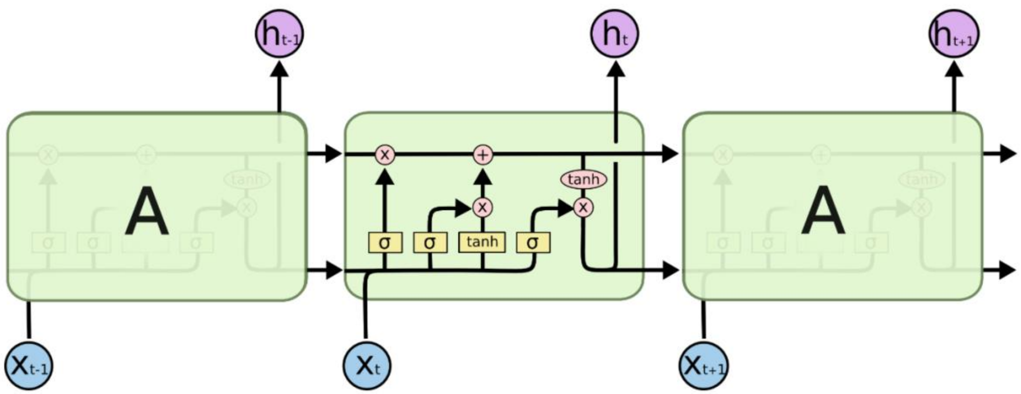
O 완료 / △미완료

Experiment Design

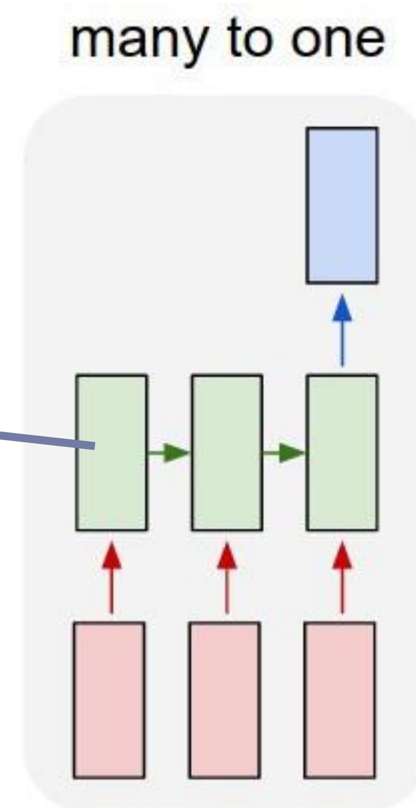
② perform sentiment analysis task

- Sentiment Analysis

~~Seq2Seq ->~~ **Vanilla LSTM Classifier**



The repeating module in an LSTM contains four interacting layers.



GREEN : LSTM cells

RED : Embedding layer

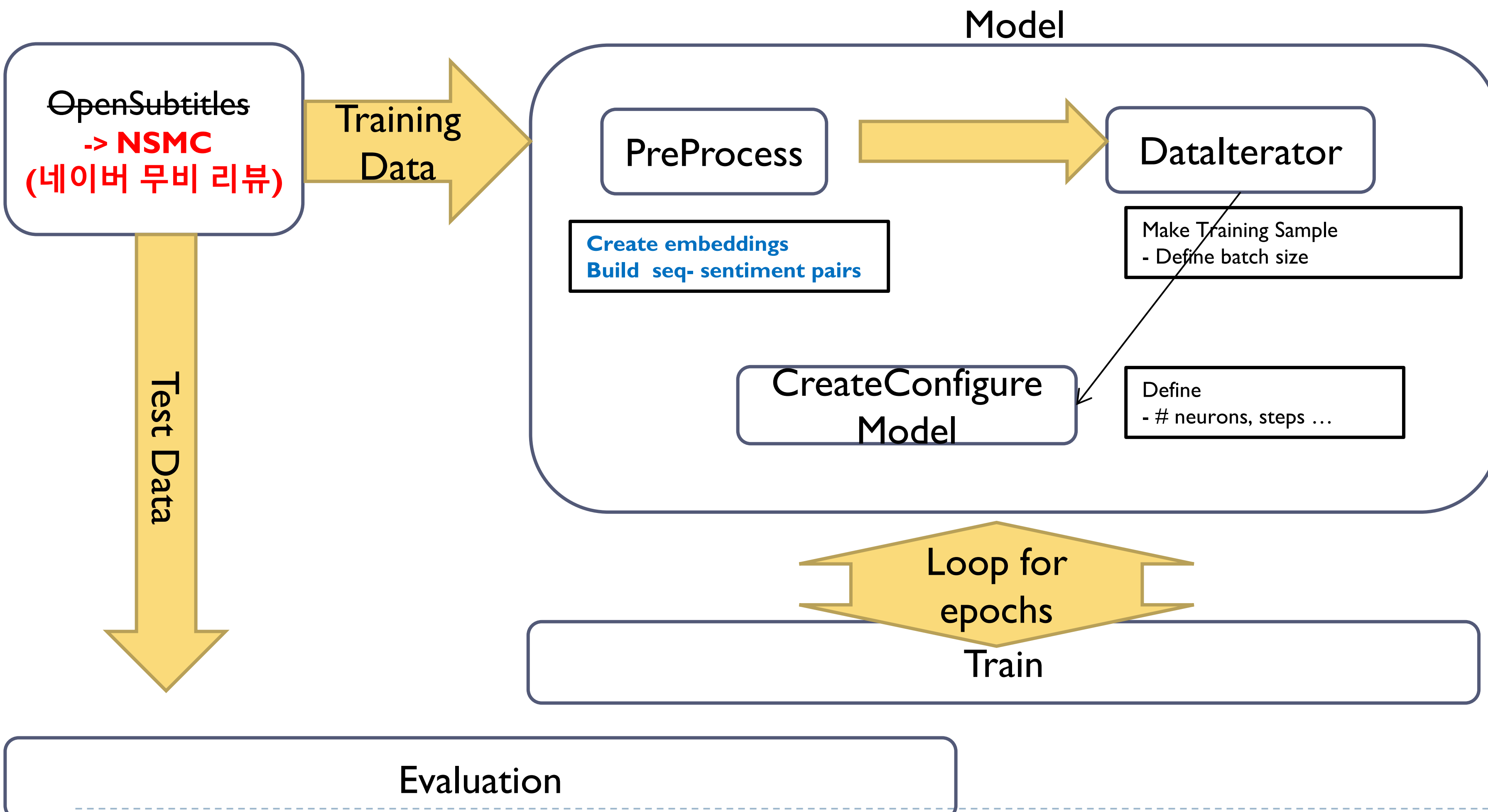
BLUE : Output layer (sigmoid activation) binary sentiment **pos**/**neg**

Development Environment

- **SNU Local server**
- Lib: Tensorflow 1.6, Keras 2.0, KoNlp, **Gensim**
- Lang: Python, C++

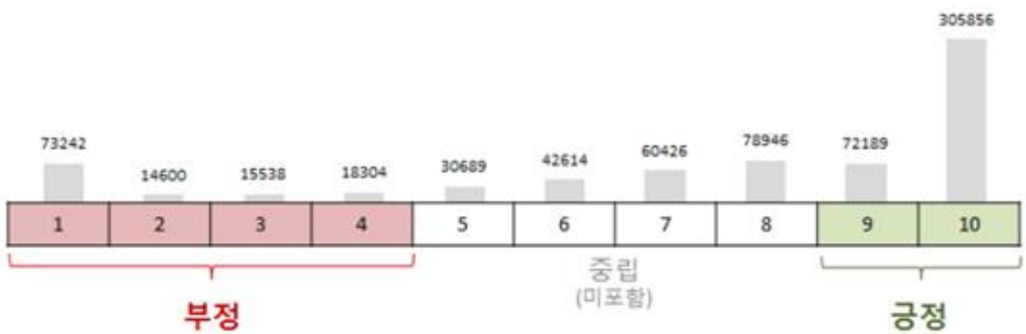
Architecture – Workflow

Sequential Diagram



Naver sentiment movie corpus v1.0

- Maas et al., 2011 데이터셋 참고
- 데이터 출처: 네이버
- 영화 당 100개의 140자평(이하 '리뷰')을 초과하지 않음
- 총 20만 개 리뷰 (수집된 64만개 중 샘플링)
  - `ratings_train.txt`: 15만, `ratings_test.txt`: 5만
  - 긍정/부정 리뷰의 비율을 동일하게 샘플링 (i.e., random guess yields 50% accuracy)
  - 중립 리뷰는 포함하지 않음



- 크기: 19MB
- URL: http://github.com/e9t/nsmc/

from : https://www.lucypark.kr/docs/2015-pyconkr/#39

Current Work

(1) 데이터 전처리

```
$ head ratings_train.txt
id      document        label
9976970 아 더빙.. 진짜 짜증나네요 목소리        0
3819312 흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나        1
10265843        너무재밓었다그래서보는것을추천한다      0
9045019 교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정       0
6483659 사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무나도 이뻐보였다  1
5403919 막 걸음마 뗀 3세부터 초등학교 1학년생인 8살용영화.ㅋㅋㅋ...별반개도 아까움.     0
7797314 원작의 긴장감을 제대로 살려내지못했다.  0
9443947 별 반개도 아깝다 욕나온다 이응경 길용우 연기생활이몇년인지..정말 발로해도 그것보단 낫겟다 납치.감금만반복반복..이드라D
7156791 액션이 없는데도 재미 있는 몇안되는 영화 1
```

각 단위의 Embedding 생성을 위해

자,모 < 음절 < 형태소 < 단어로 분리

형태소 분석기: Mecab-ko / Kkma / Twitter 중 택 1

Current Work

- (2) 임베딩 생성

Hyperparameters

embedding dimension	50, 100, 300 ,500 ,1000
window size	2, 5, 7, 10
minimum count	10, 20, 50, 100
minimum n-gram(FastText)	**1, 2, 3**

* 나머지 Epoch, negative sampling rate 등의 hyper parameter은 원 논문에서 밝힌 수치를 사용

총 6(+a)개의 다른 embedding을 구함

- Word2vec – word, morpheme
- Glove – word, morpheme
- FastText – word/morpheme syllable
- +extra jamo , different morpheme parser, n gram

Current Work

- (3)학습 배치 생성

Tokens to Integers list

['아', '더빙', '..', '진짜', '짜증', '나네', '요', '목소리']

→ [182, 11, 132, 22, 17, 98, 16, 52]

Integers to numpy arrays – 1 hot encoding

→ [[0,0,0,0,0,…1,0,0,0],[0,0,0,0,0,…,1,0,0] …]

Padding

→50(truncate above) or to max sequence length

Labels – 1 hot encoding

[[0, 1]], [1, 0] …]

Current Work

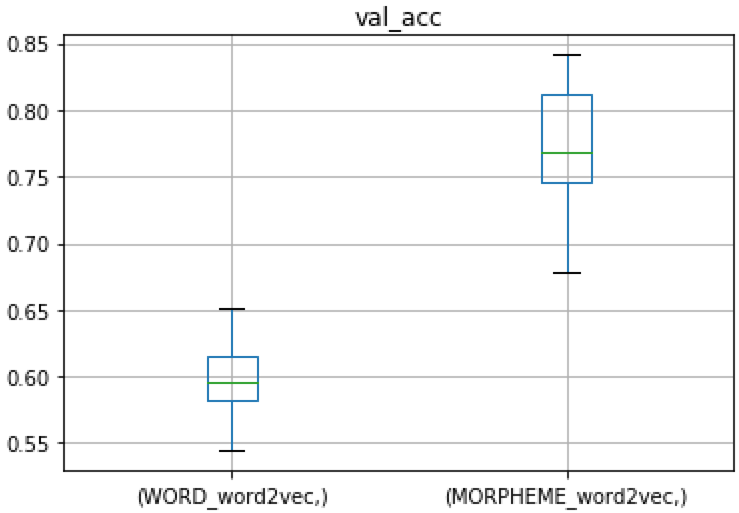
- (4) 훈련

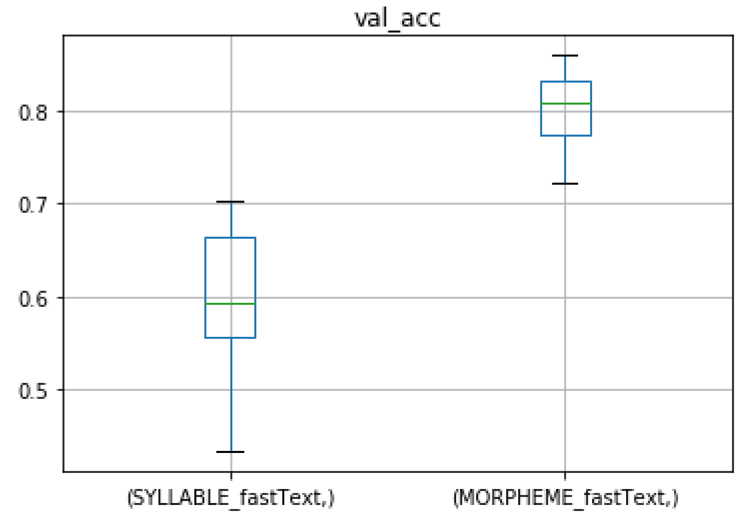
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50)	0
embedding_1 (Embedding)	(None, 50, 300)	88354800
lstm_1 (LSTM)	(None, 300)	721200
dense_1 (Dense)	(None, 2)	602

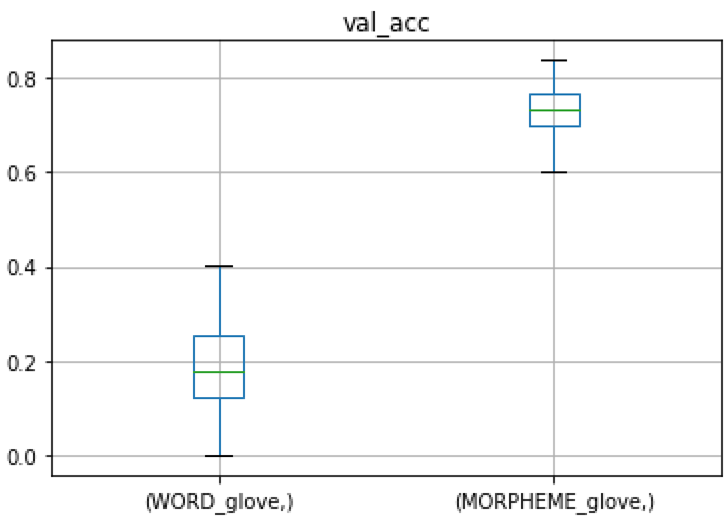
Total params: 89,076,602
Trainable params: 721,802
Non-trainable params: 88,354,800

Optimizer: adam, **Loss** : binary_crossentropy, **EarlyStopping** : 5 (patience level)

Preliminary Result – word vs morpheme

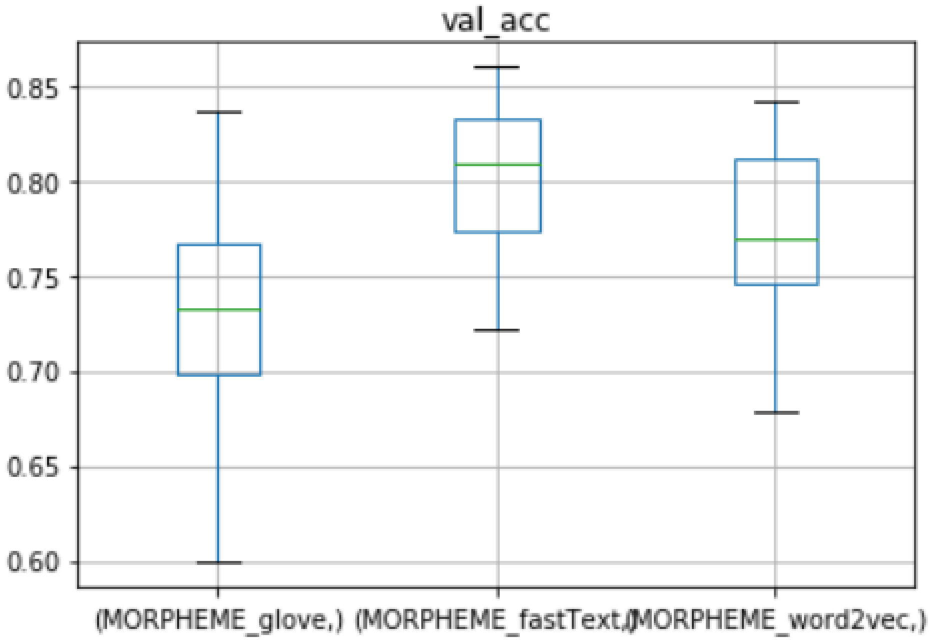






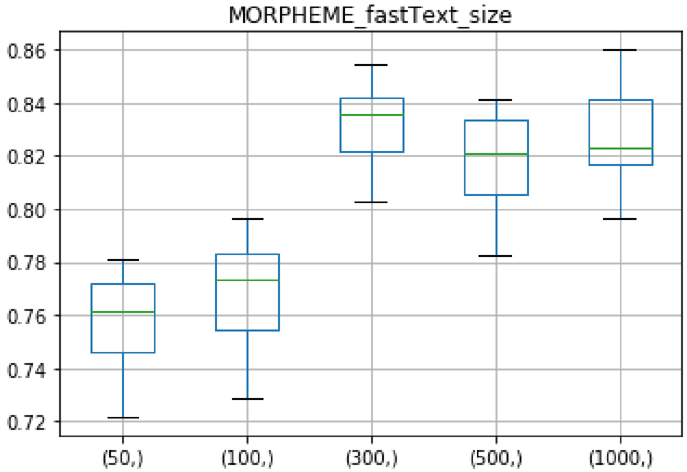
모든 모델에서 형태소 단위의 입력이 성능이 더 좋음

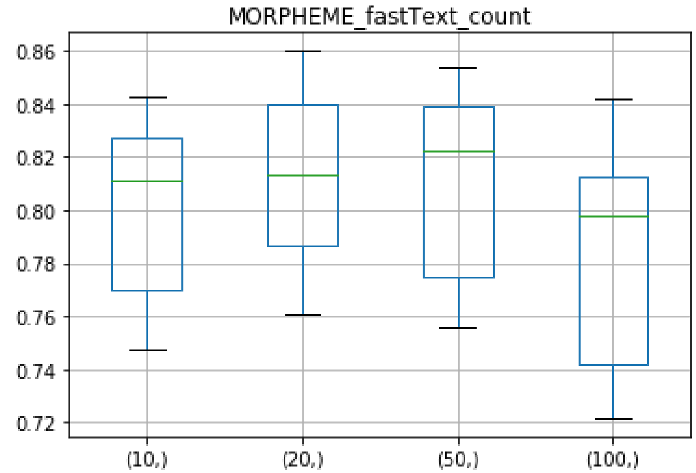
Preliminary Result – between models

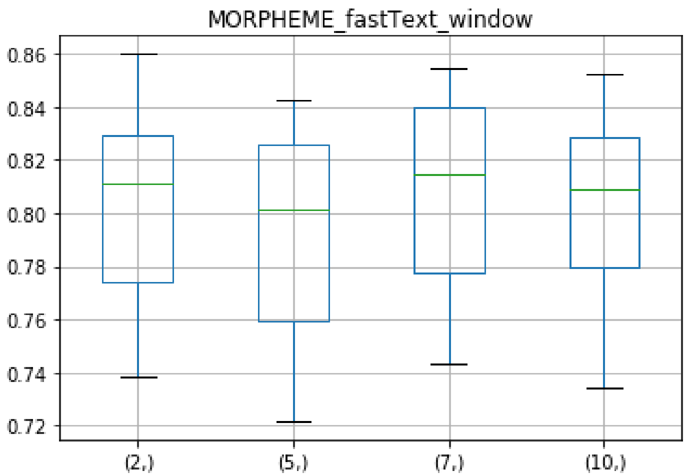


형태소 단위의 입력에서는
fastText > w2vec(sgns) ~ glove¶

Preliminary Result – compare params



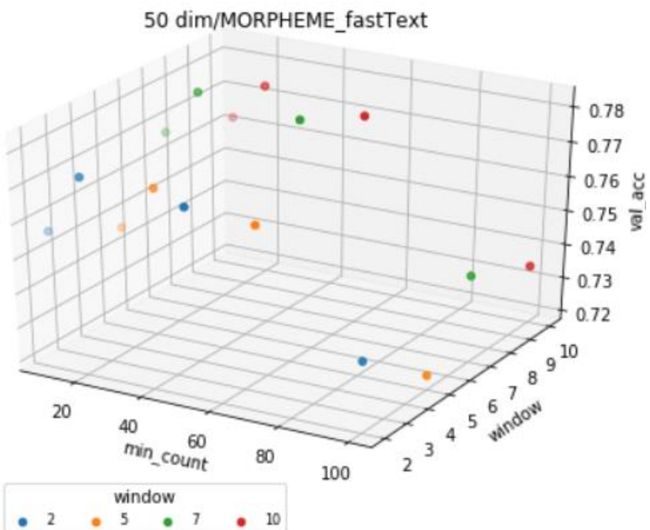


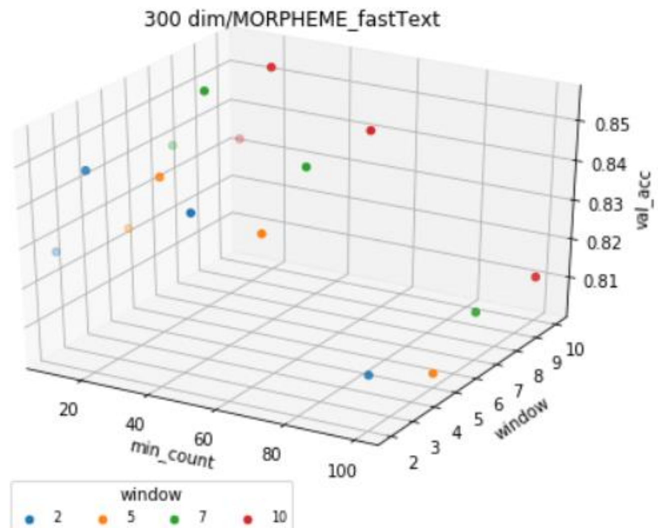


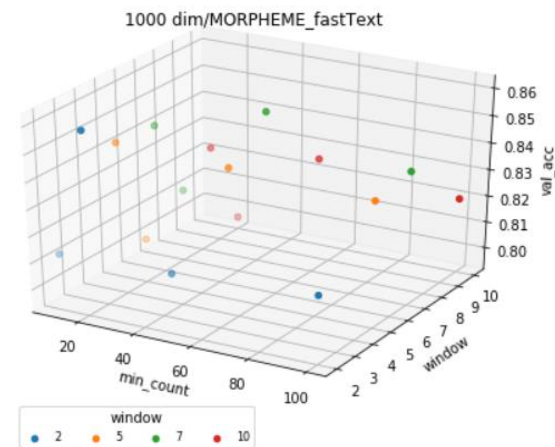
size파라메터가 모델의 정확도에 가장 영향을 많이주었다¶

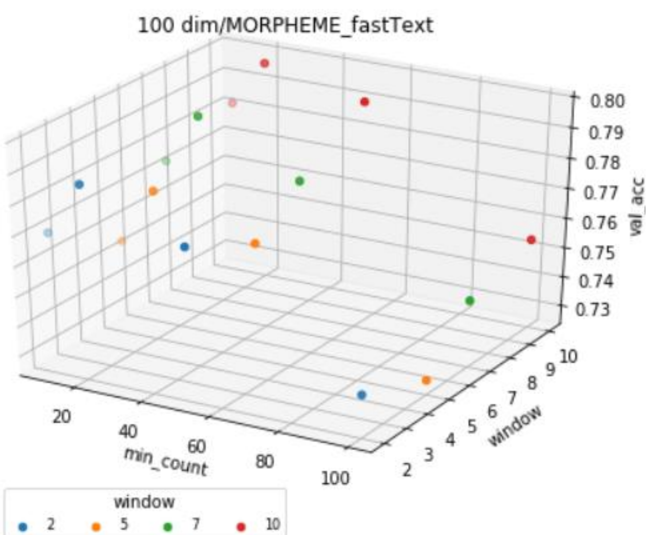
window 나 minimum count 파라메터의 조합에 따른 성능 비교는 다음에서 확인할 수 있다

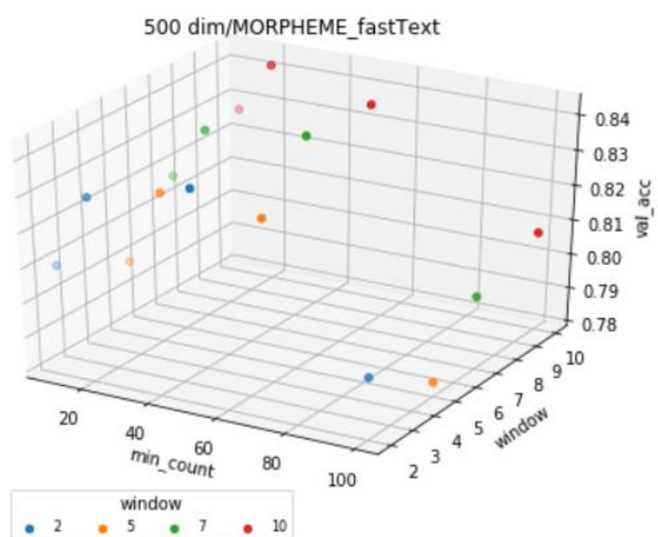
Preliminary Result – window-min_count



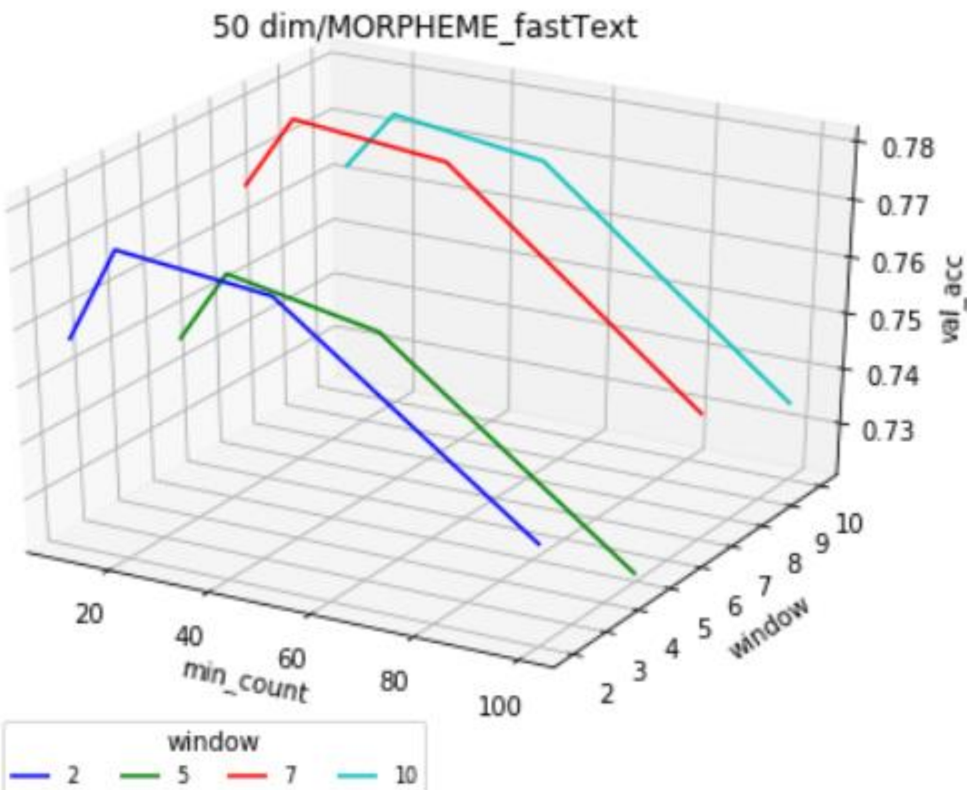






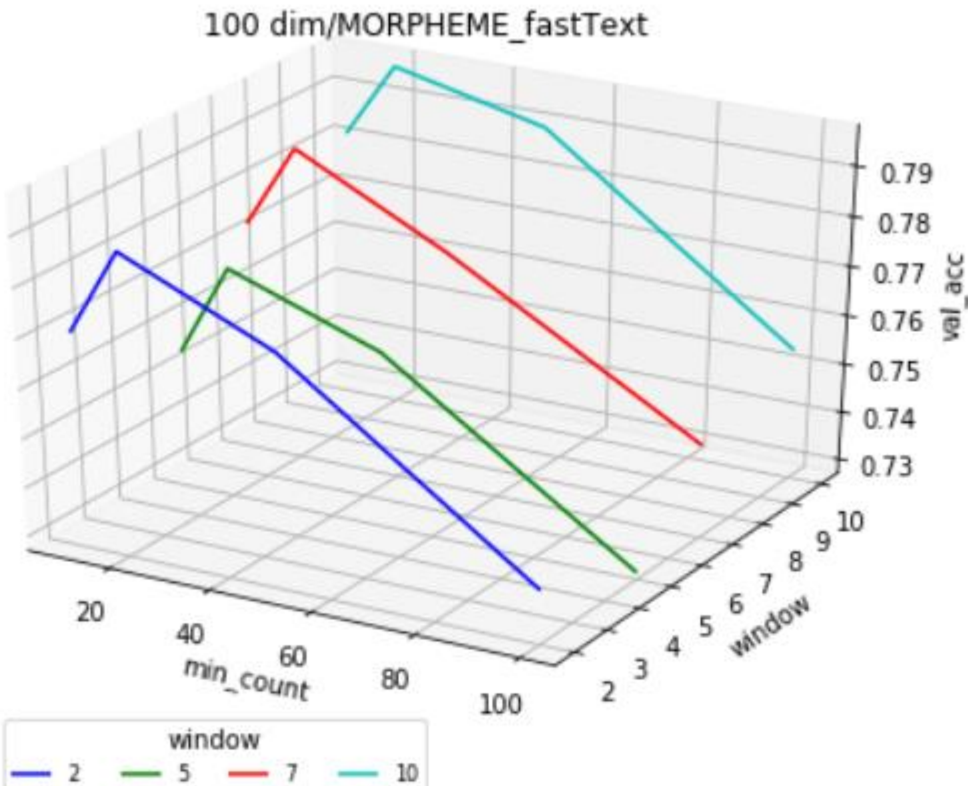


Preliminary Result – groupby window



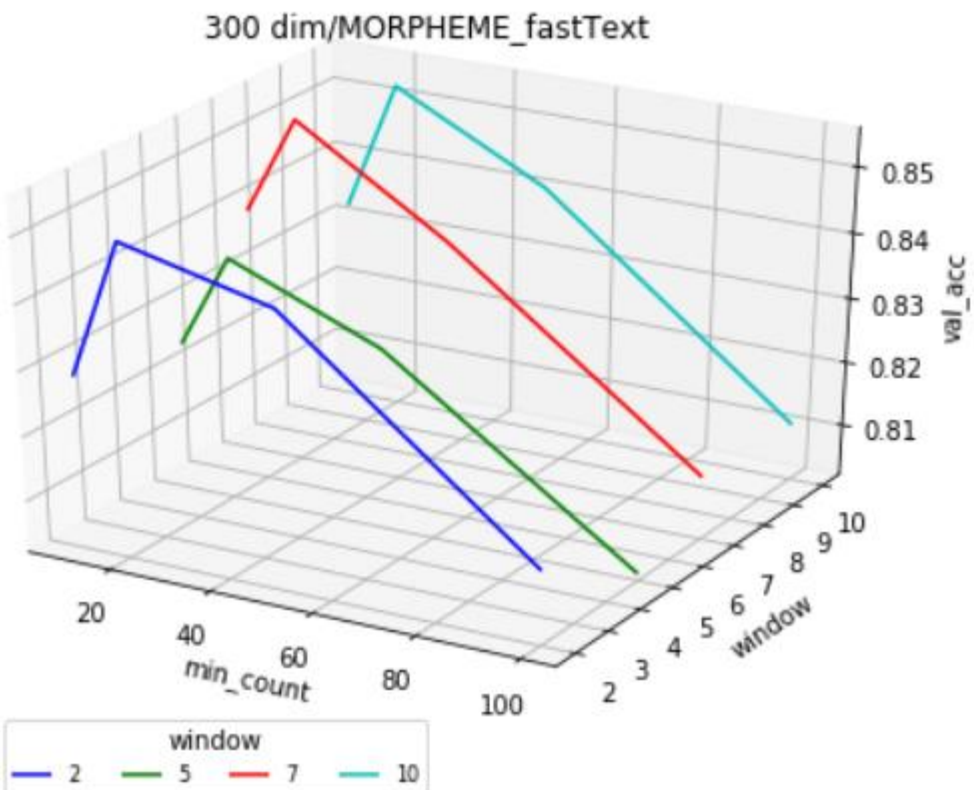
50 : [5 10 2 7]
100 : [5 2 7 10]
300 : [5 2 10 7]
500 : [5 7 2 10]
1000 : [10 5 7 2]

Preliminary Result – groupby window



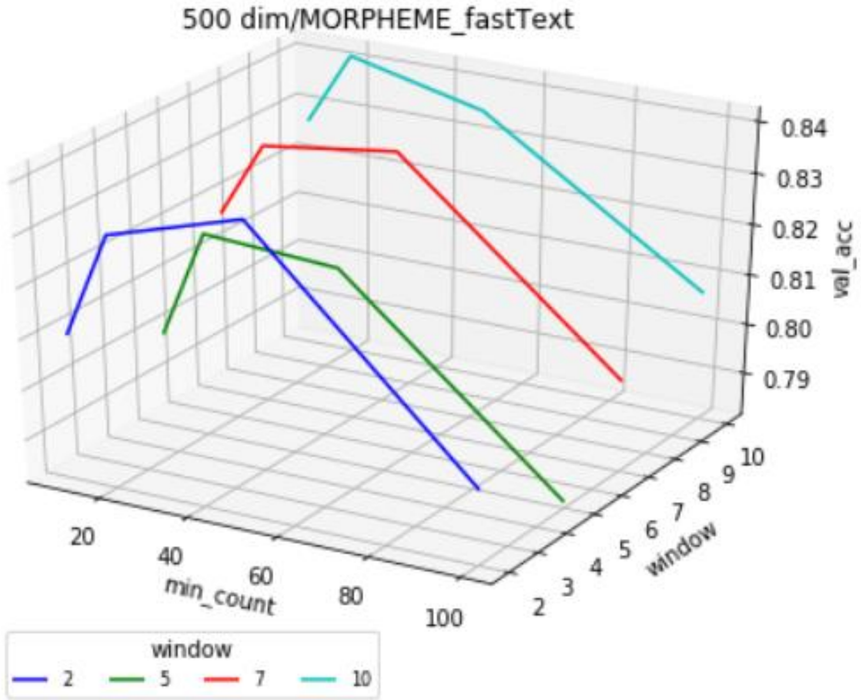
50 : [5 10 2 7]
100 : [5 2 7 10]
300 : [5 2 10 7]
500 : [5 7 2 10]
1000 : [10 5 7 2]

Preliminary Result – groupby window



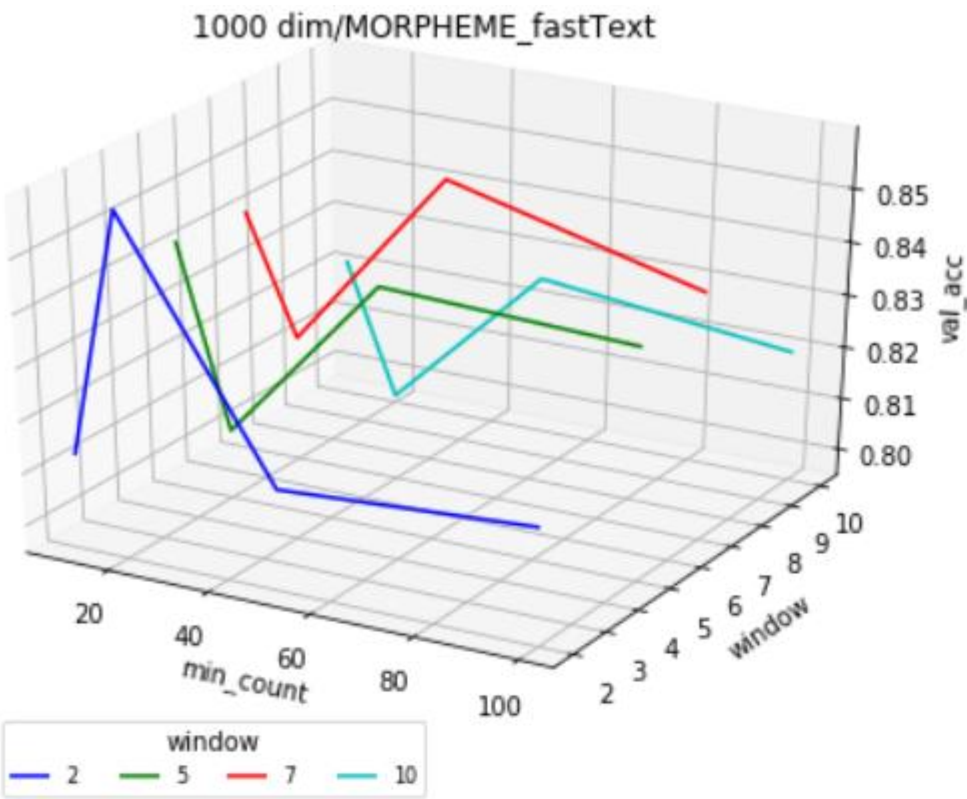
50 : [5 10 2 7]
100 : [5 2 7 10]
300 : [5 2 10 7]
500 : [5 7 2 10]
1000 : [10 5 7 2]

Preliminary Result – groupby window



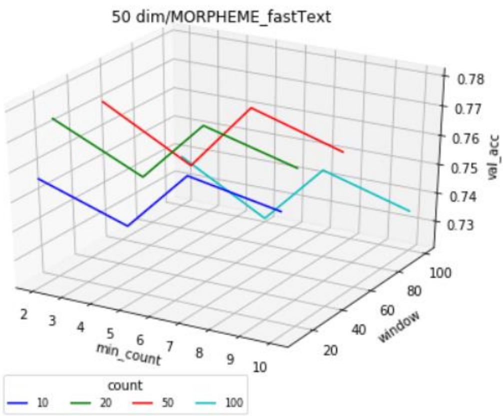
50 : [5 10 2 7]
100 : [5 2 7 10]
300 : [5 2 10 7]
500 : [5 7 2 10]
1000 : [10 5 7 2]

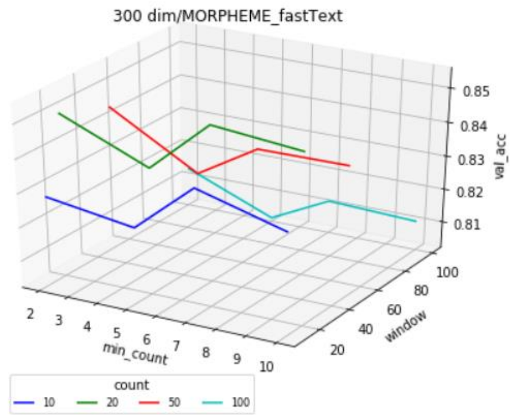
Preliminary Result – groupby window

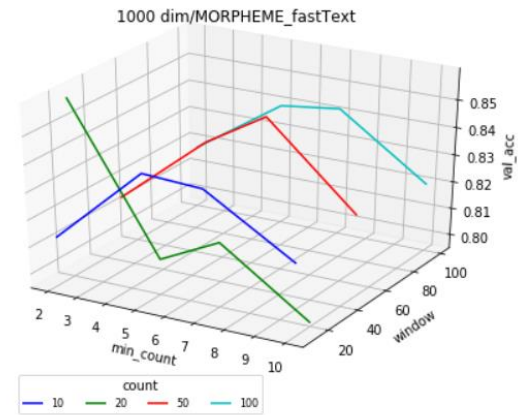


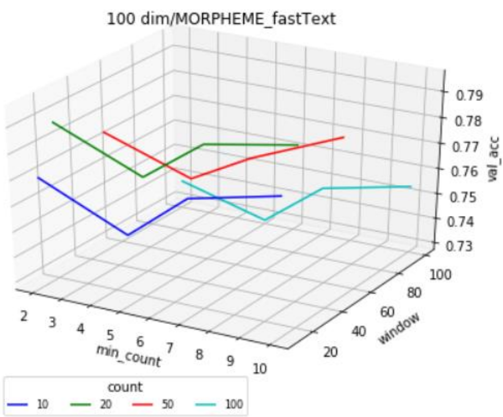
50 : [5 10 2 7]
100 : [5 2 7 10]
300 : [5 2 10 7]
500 : [5 7 2 10]
1000 : [10 5 7 2]

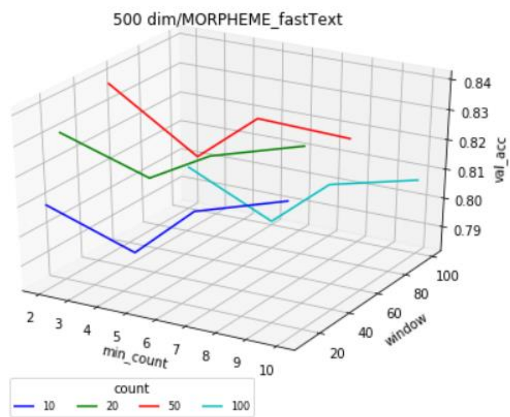
Preliminary Result – groupby count





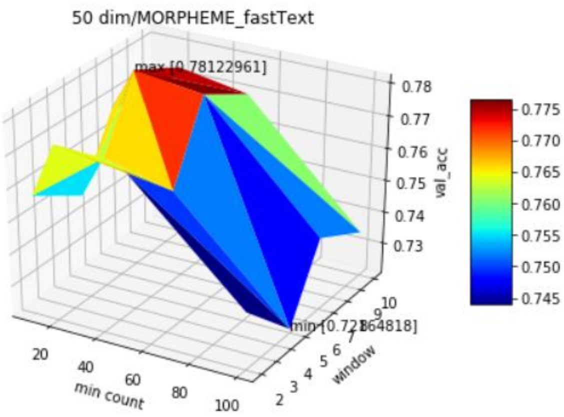


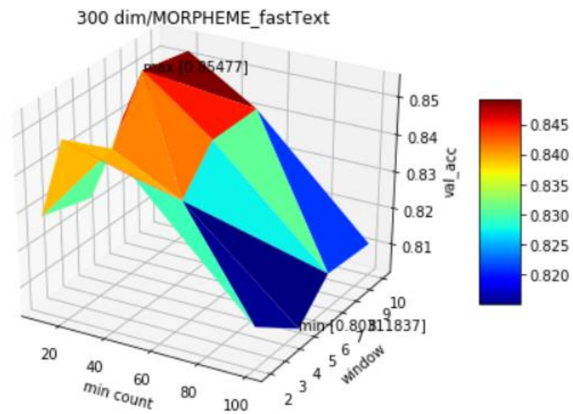


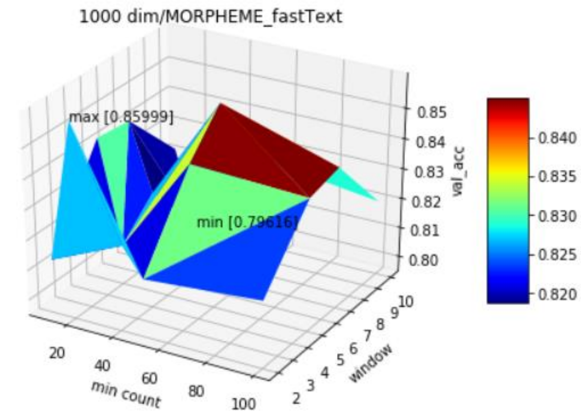


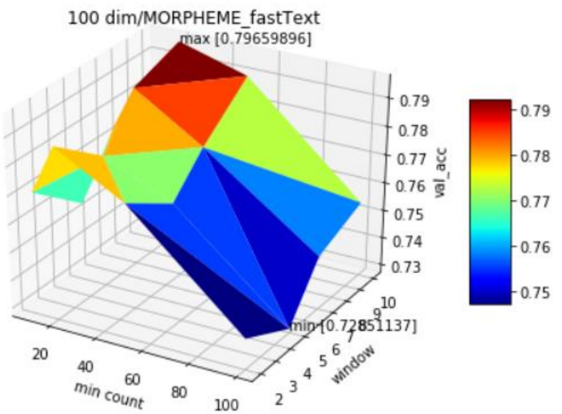
50 : [100 10 50 20]
100 : [100 10 50 20]
300 : [100 10 50 20]
500 : [100 10 50 20]
1000 : [100 10 50 20]

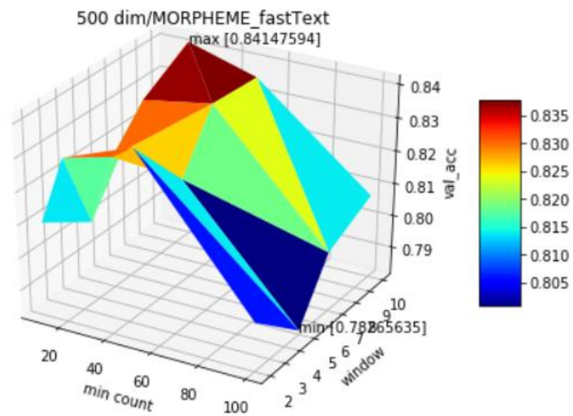
Preliminary Result – surf plot





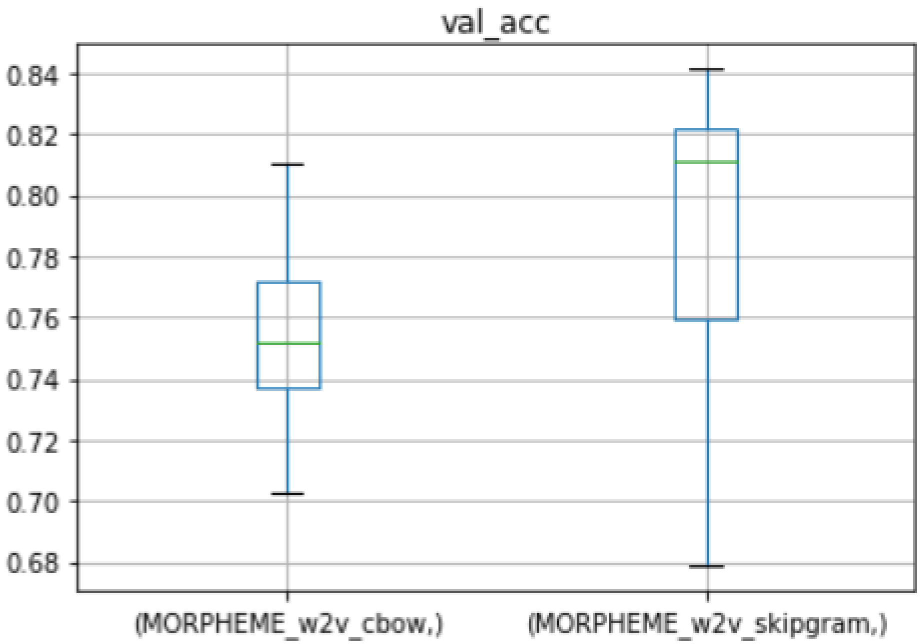






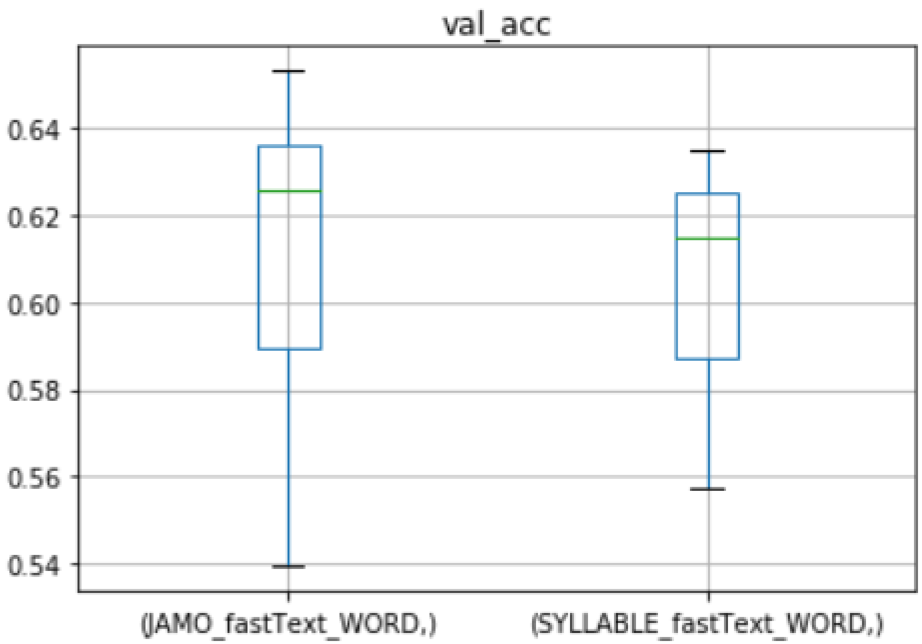
300 dim/MORPHEME_fastText

Preliminary Result – SkipGram vs CBOW



Word2vec 에서는
SkipGram 이 CBOW보다 더 우세했다.
* 둘 모두 loss함수는 negative sampling 방식이다.

Preliminary Result – Jamo vs Syllable



fastText에서는 ngram의 단위로 자모 단위가 음절 단위보다 성능이 더 좋았다.

- Skipgram negative sampling
- Input unit word

Preliminary Result – Top 5 params

	type	model	window	size	count	acc	loss	val_acc	val_loss
333	WORD	word2vec	7	300	10	0.377015	0.900680	0.650707	0.385647
339	WORD	word2vec	10	300	10	0.380762	0.883067	0.636348	0.439269
341	WORD	word2vec	7	300	50	0.351410	0.879842	0.635897	0.375007
342	WORD	word2vec	7	300	20	0.617167	0.614238	0.632890	0.540539
343	WORD	word2vec	7	1000	10	0.358466	0.886352	0.632841	0.365796

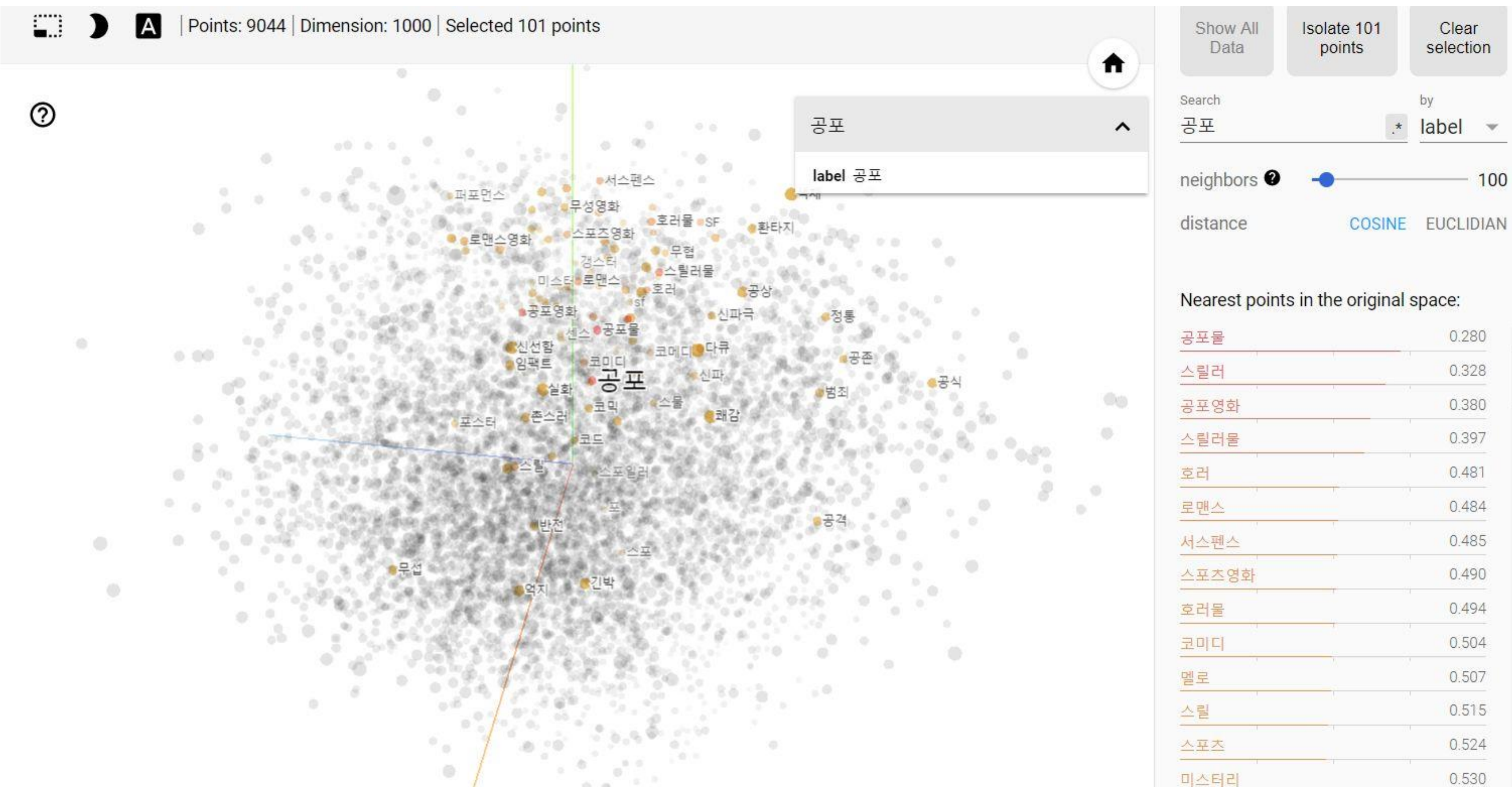
	type	model	window	size	count	acc	loss	val_acc	val_loss
12	MORPHEME	word2vec	10	500	20	0.931570	0.374700	0.841400	0.563303
17	MORPHEME	word2vec	10	500	10	0.831208	0.846291	0.839851	0.663142
22	MORPHEME	word2vec	10	500	50	0.824999	0.856095	0.837884	0.729713
25	MORPHEME	word2vec	5	1000	10	0.820471	0.830271	0.835784	0.660463
31	MORPHEME	word2vec	7	1000	10	0.962463	0.357063	0.831170	0.600407

Preliminary Result – Top 5 params

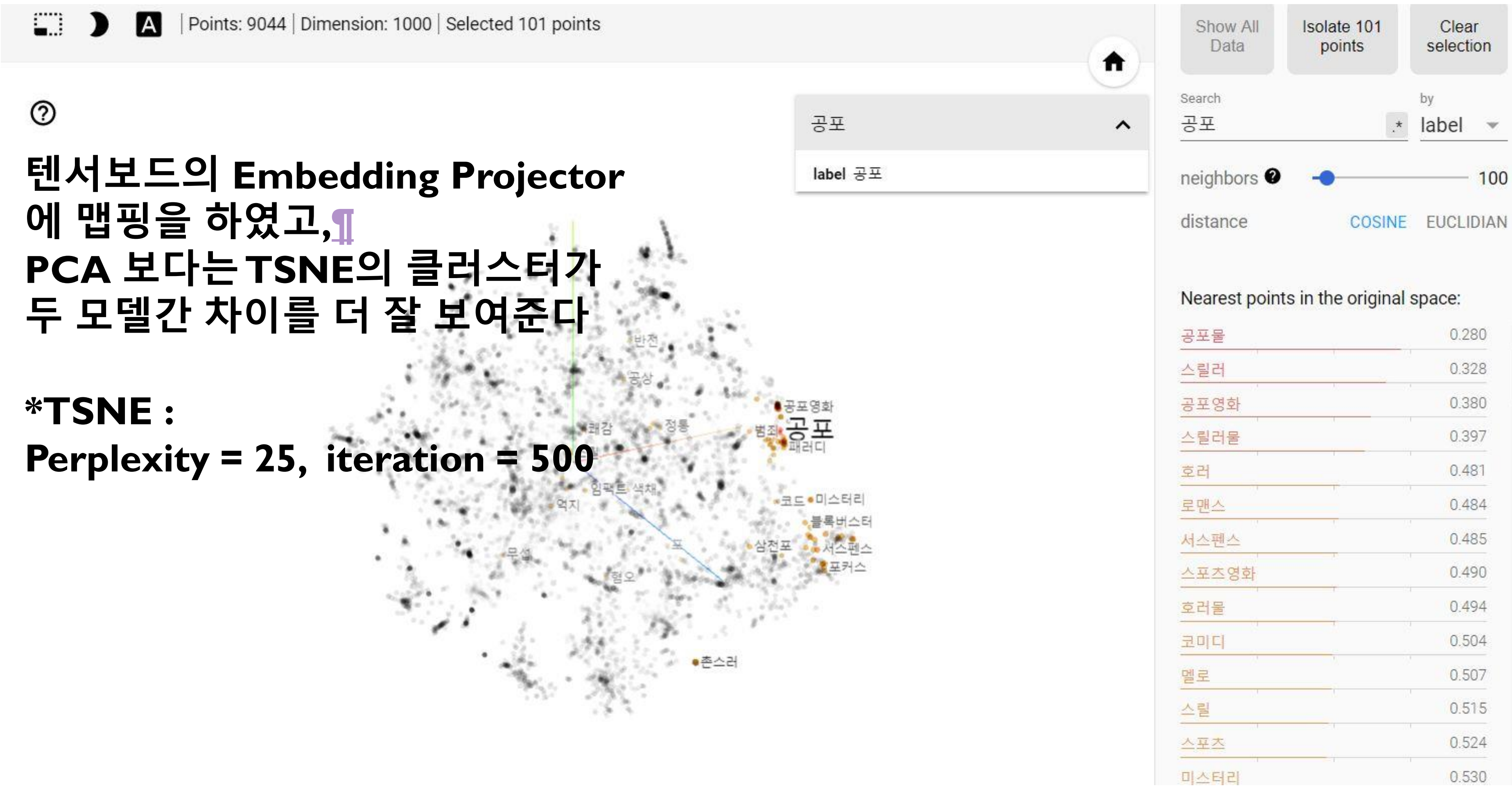
	type	model	window	size	count	acc	loss	val_acc	val_loss
23	MORPHEME	glove	7	1000	10	0.600841	1.046637	0.837040	0.774622
27	MORPHEME	glove	7	500	10	0.770527	0.805446	0.833323	0.770552
52	MORPHEME	glove	10	1000	10	0.656234	1.051976	0.822339	0.777445
68	MORPHEME	glove	10	500	10	0.709046	0.767489	0.818054	0.779282
90	MORPHEME	glove	5	1000	10	0.693206	0.746265	0.808366	0.767530

	type	model	window	size	count	acc	loss	val_acc	val_loss
1	MORPHEME	fastText	2	1000	20	0.857537	0.455650	0.859990	0.831212
4	MORPHEME	fastText	7	300	20	0.928033	0.407341	0.854770	0.589227
5	MORPHEME	fastText	7	1000	50	0.875044	0.790955	0.853934	0.590042
6	MORPHEME	fastText	10	300	20	0.926173	0.417405	0.852230	0.562615
7	MORPHEME	fastText	2	300	20	0.930363	0.405508	0.850180	0.583748

Preliminary Result – PCA



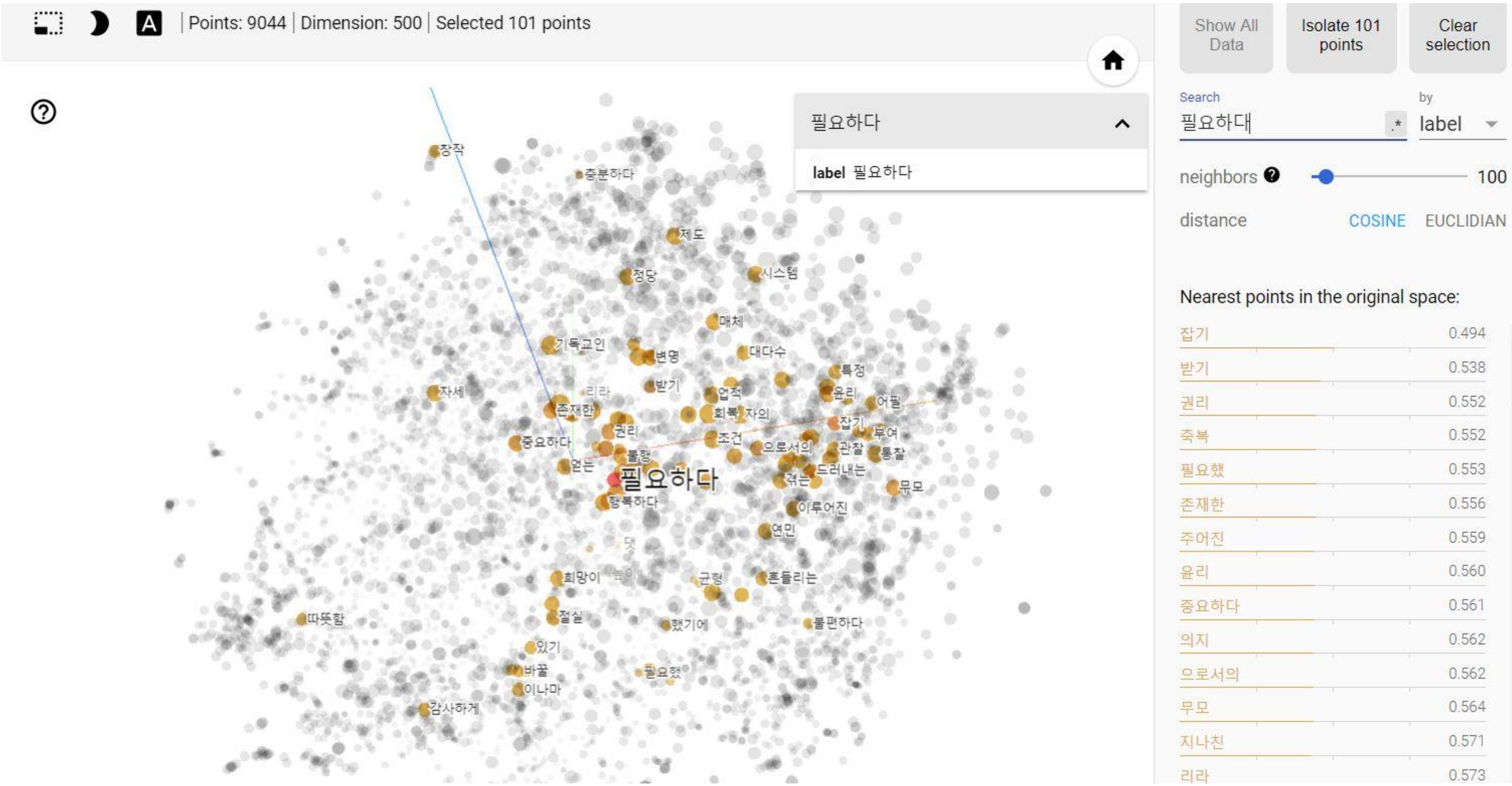
Preliminary Result – TSNE



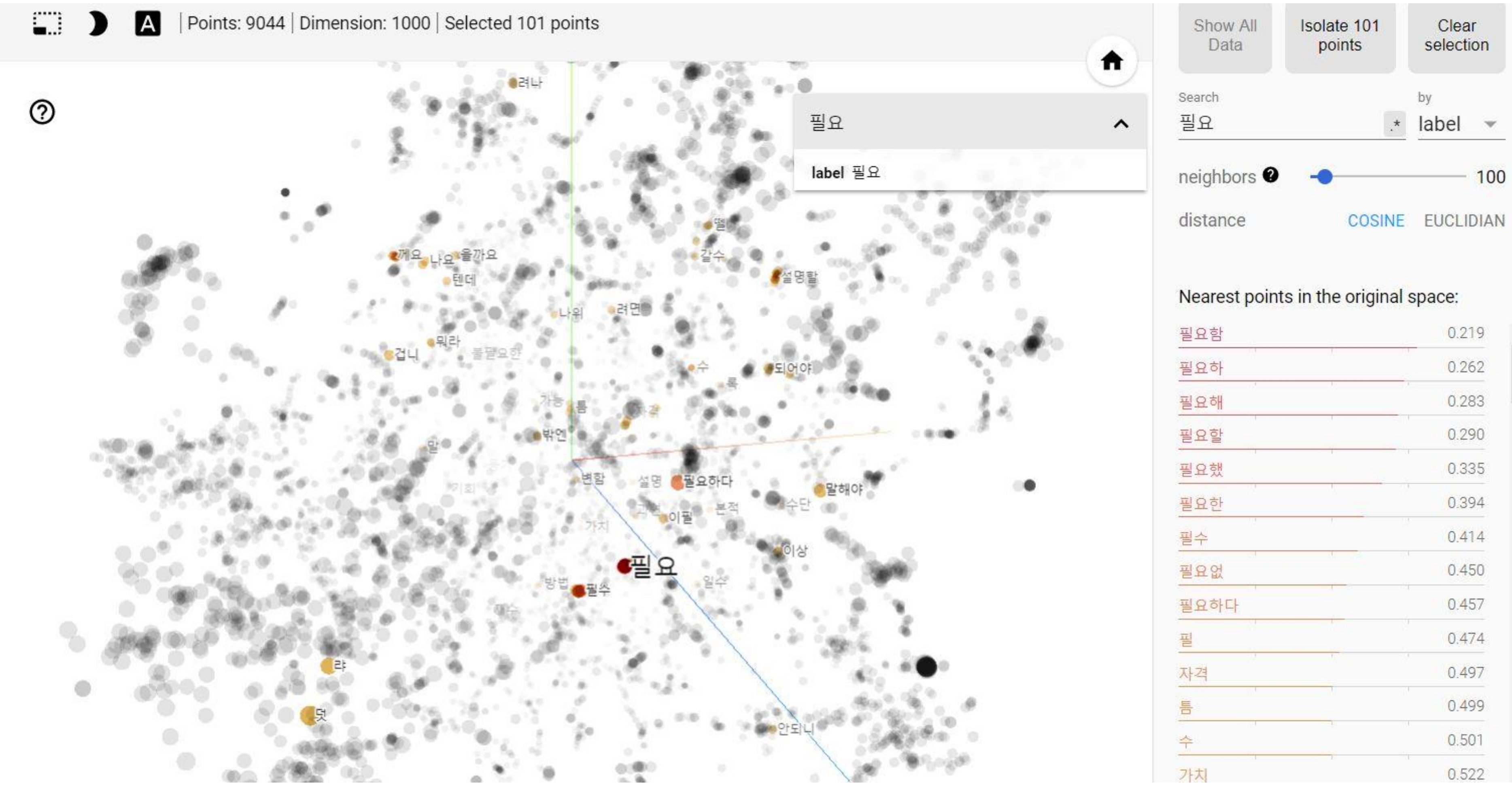
텐서보드의 Embedding Projector 에 맵핑을 하였고,
PCA 보다는 TSNE의 클러스터가 두 모델간 차이를 더 잘 보여준다

***TSNE :**
Perplexity = 25, iteration = 500

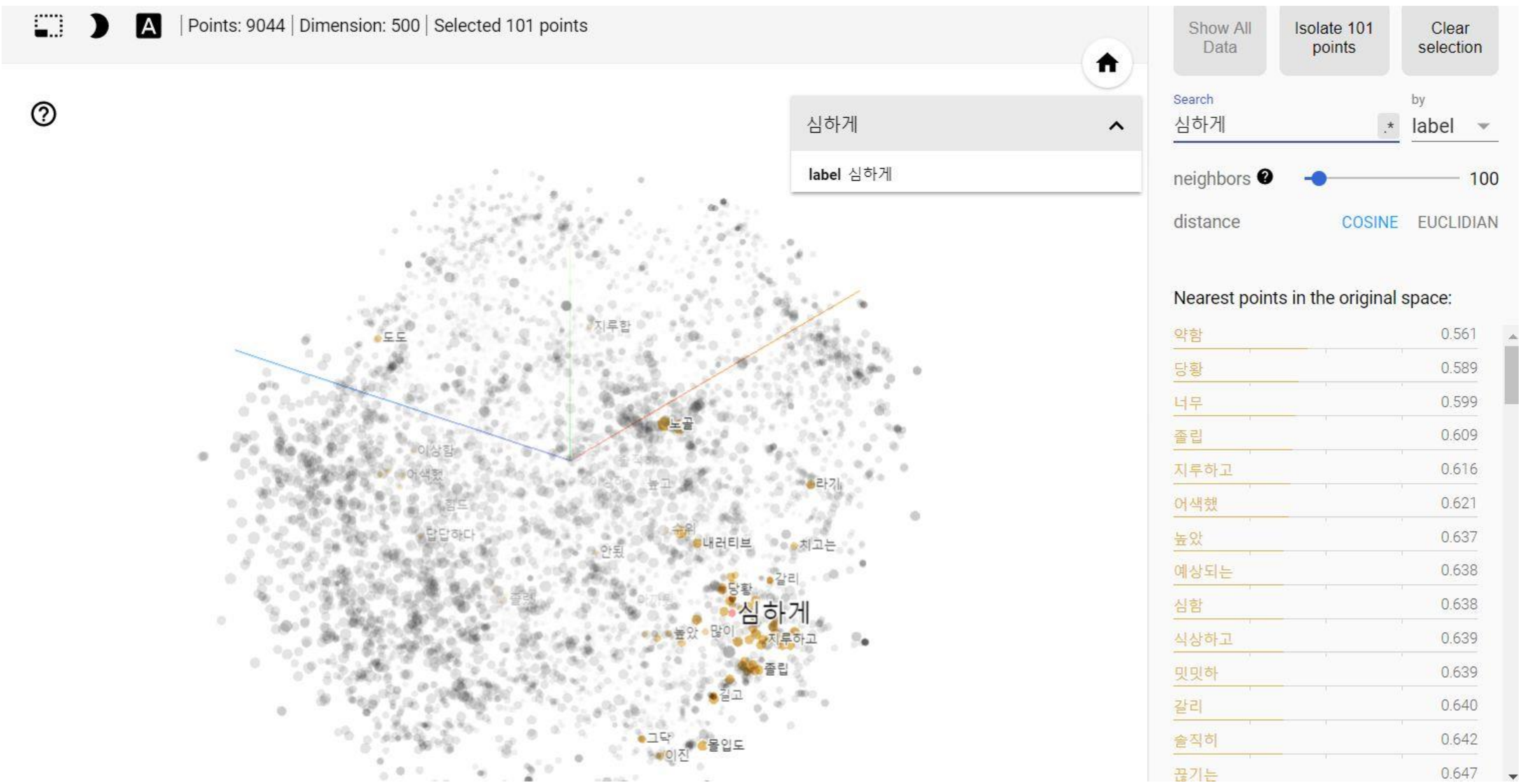
Preliminary Result – word2vec vs fastText



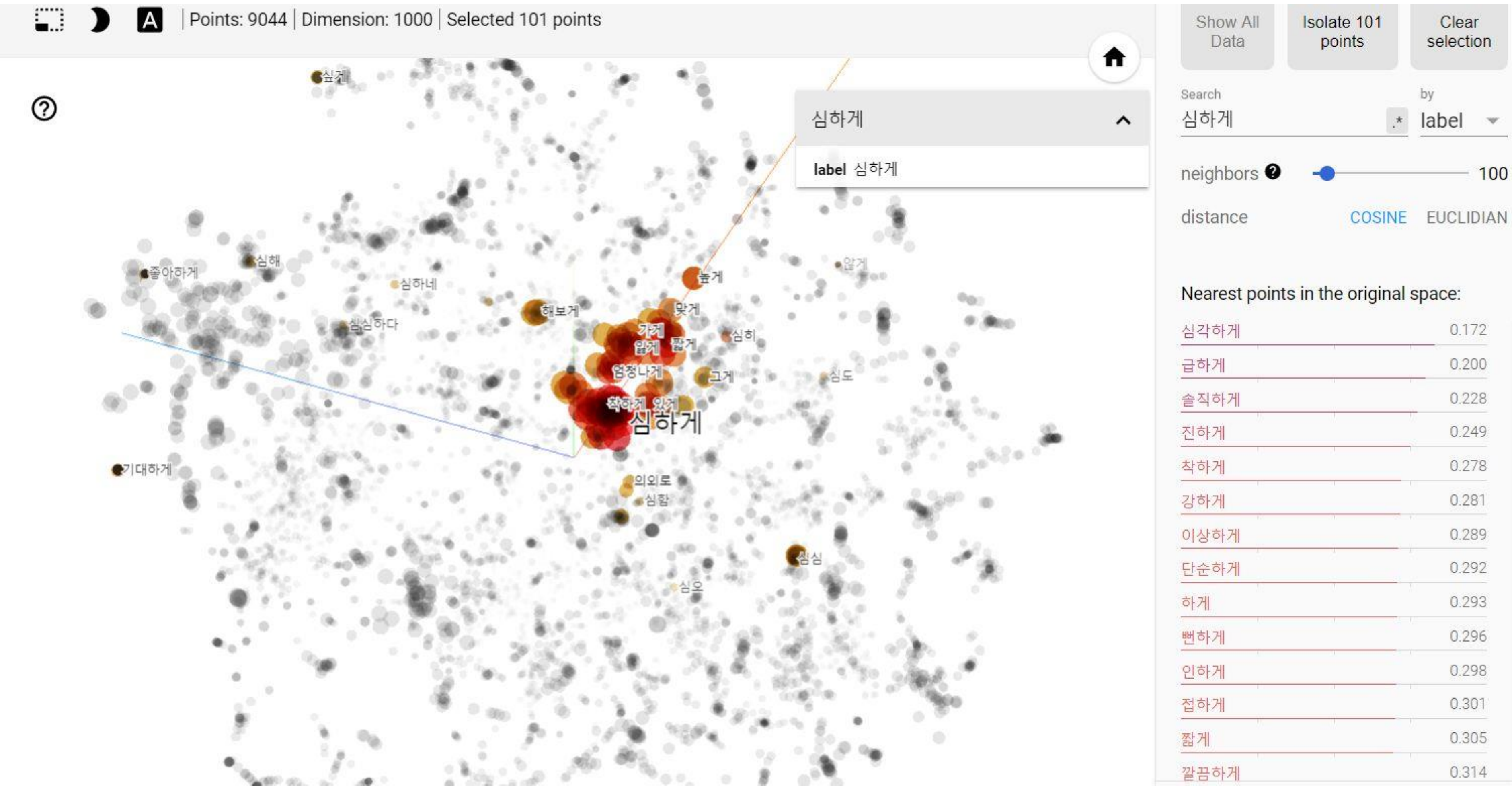
Preliminary Result – word2vec vs fastText



Preliminary Result – word2vec vs fastText



Preliminary Result – word2vec vs fastText



Preliminary Result – word2vec vs fastText

word2vec

$$\sum_{t=1}^{T} \sum_{c \in C_t} logp(w_c|w_t)$$

$$p(w_c|w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^{W} e^{s(w_t, j)}}$$

fastText

$$s(w, c) = \sum_{g \in g_w} z_g^T v_c$$

fastText 모델에 따르면 target word와 gram을 공유하는
단어의 벡터도 변하기때문에
원래의 윈도우에 포함되어 있지 않은 단어일지라도
학습가능하다.

참고문헌

- Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." 2016.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." 2013.
- Naili, Chaibi, and Ben Ghezala. "Comparative Study of Word Embedding Methods in Topic Segmentation." Procedia Computer Science 112 (2017): 340-49.
- Omer Levy, Yoav Goldberg, and Ido Dagan. “Improving distributional similarity with lessons learned from word embeddings.” TACL. 2015a.
- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. “Glove: Global Vectors for Word Representation.” In EMNLP,vol. 14,pp. 1532-43. 2014
- Shi, Tianze, and Zhiyuan Liu. "Linking GloVe with Word2vec." 2014.
- 최상혁 외, 제28회 한글 및 한국어 정보처리 학술대회 논문집 (HCLT), 2016.

Schedule

내용	3월			4월				5월				6월	
데이터 전처리	○	○	○										
네트워크 정의			○	○	○								
훈련 단계					○	○	○	○					
평가 단계							○	○					
코드 리뷰 /데모 계획						○	○	○	○				
논문 작성				○	○	○	○	○	○	○	○	○	○
논문 발표/수정									○	○	○	○	○

감사합니다

이한결(miroblog@snu.ac.kr)