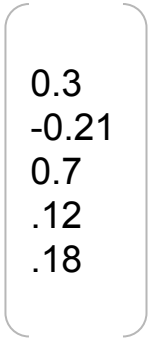
한국어 형태적 특징을 반영한 단어 임베딩 모델(Skipgram, Glove, Fasttext)의 통합적 고찰

-매개변수 조절과 감정분석에 적용한 성능비교

Background

Word Embedding ?

KING =



Analogy ?

King : Queen = Man : ?

? = vec(King) - vec(Queen) + vec(Man)

Similarity?

Top5(Superman)

-> Hero, Batman, Ironman, X-men, Spiderman

Embedding Methods

Prediction Based

Trying to predict the **word** from its **neighbors**

(e.g.) NPLM, skipgram, cbow, fastText ...

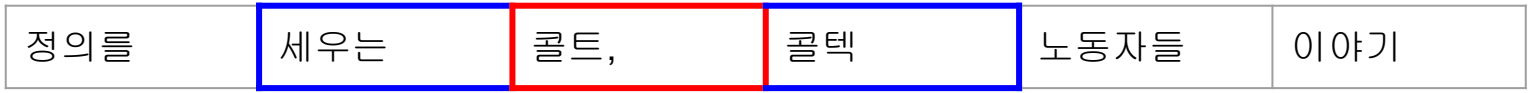
Count Based

How often the same word **co-occurs** with its **neighbor** words in a large text corpus

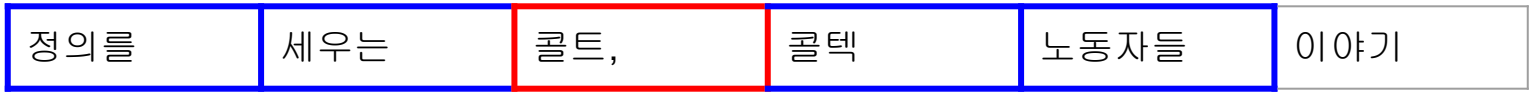
(e.g.) LSA , GloVe ...

Local Context Window

Window size : 1



\Window size :2



Red - Target Word / Blue - Context Word

Collect - D

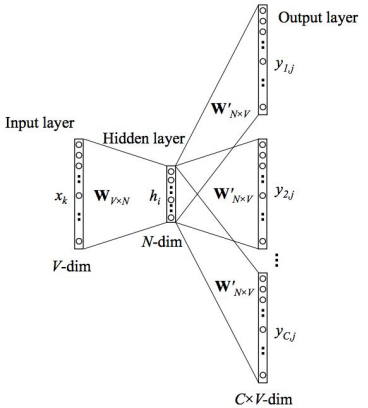
D: All (**word**, **context**) pair in the corpus

Sliding Window (Size=1)	Target Word	Context Word
[정의를 세우는]	정의를	세우는
[정의를 세우는 콜트]	세우는	정의를, 콜트
[세우는 콜트, 콜텍]	콜트	세우는, 콜텍
[콜트, 콜텍, 노동자들]	콜텍	콜트, 노동자들
[콜텍, 노동자들, 이야기]	노동자들	콜텍, 이야기
[노동자들, 이야기]	이야기	노동자들

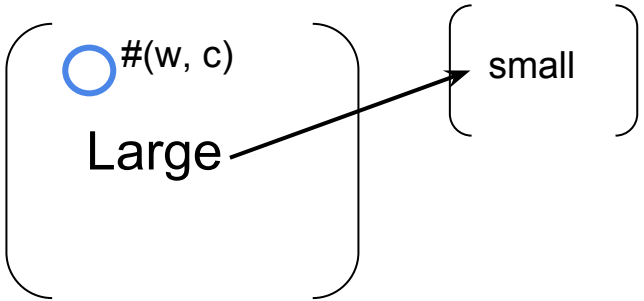
Prediction Based

Given X , Predict Y or

Given Y, Predict X



Count Based



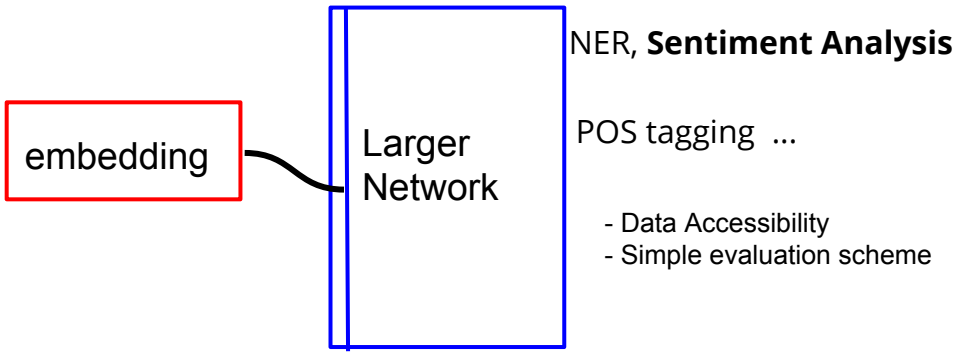
So...?

Evaluation ?

Intrinsic Evaluation : Similarity or Analogy ...

- 성능이 data set 마다 다르게 측정됨

Extrinsic Evaluation : Task Based



Korean ?

Base Input Unit : Word

- 단어당 평균 빈도수가 유지되기 힘듬, 노이즈로 작용

Simple Solution

Word → Morpheme

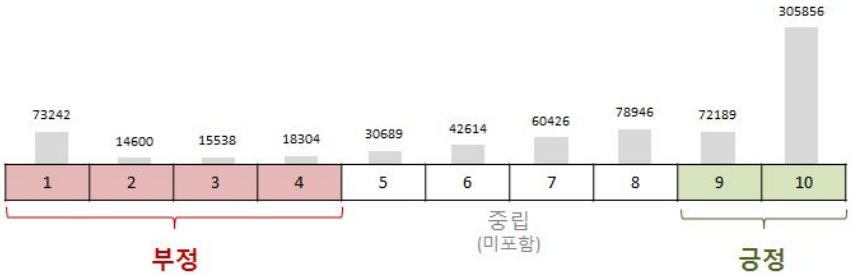
***(+ α* fastText :** *Syllable → Grapheme)*

→ **How to tune parameters?**

DataSet

NSMC(Naver Sentiment Movie Corpus)

- 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



Movie Review Example

Id	Document	Label
8112052	어릴때보고 지금다시봐도 재밌어요ㅋㅋ	1
9125596	가면갈수록 내용이 개판이다. 많은걸 집어넣으려다망함	0

Experiment?

Step 1 — Create Embedding

Step 2 — Create Training Samples

Step 3 — Define Network

Step 4 — Train & Validate

1. Create Embeddings

Input Unit

모델 / 입력단위	CBOW	SkipGram	Glove	FastText	
				음절	자음,모음(grapheme)
단어	["그는", "학생이다"]			[["그","는"], ["학","생","이","다"]	[["ㄱ, ㅡ, ㄴ", "ㄴ,ㅡ,ㄴ"], ["ㅎ","ㅏ","ㄱ"...]
형태소	["그", "는", "학생", "이다"]			[["그"], ["는"], [["학","생"], [["이","다"]]]	[["ㄱ, ㅡ, ㄴ", "ㄴ,ㅡ,ㄴ"], ["ㅎ","ㅏ","ㄱ"...]

*Morpheme Analysis : **Twitter** Morpheme Analyzer

Parameter

Dimension Size : 50 , 100 ,300, 500, 1000
Window : 2, 5, 7, 10
Minimum count : 10, 20, 50, 100

_____
Others : Default

Parameter Variation * # Input Variation + # partial result
= 80 * 8 + **α**

Experiment?

Step 1 — Create Embedding

Step 2 — Create Training Samples

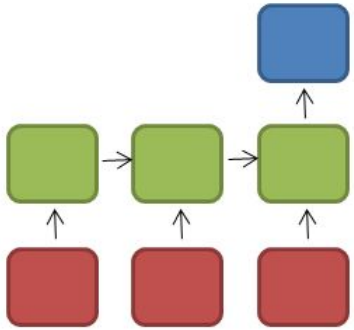
Step 3 — **Define Network**

Step 4 — **Train & Validate**

2. Create Training Samples

(1)	K_{word}^{raw}	['아', '더빙', '진짜', '짜증', '나네', '요', '목소리']
(2)	$K_{word}^{integer}$	[182, 132, 22, 17, 98, 16, 52]
(3)	K_{word}^{1-hot}	[[0,0,0,0,0,...1,0,0,0],[0,0,0,0,0,...,1,0,0] ...]
(4)	zero_padding(K_{word}^{1-hot})	[[0,0,0,0,0,...1,0,0,0],[0,0,0,0,0,...,1,0,0] ... **[0,0, ...,0,]**]
(5)	labels – 1hot encoding	[[0, 1]], [1, 0] ...]

3. Define Network



Dense layer

LSTM Layer

Embedding Layer

Experiment?

Step 1 — Create Embedding

Step 2 — Create Training Samples

Step 3 — Define Network

Step 4 — **Train & Validate**

4. Train & Validate

- # training : 15k, validation : 5k
- batch size : 128
- maximum epoch : 20
- Early Stopping: patience 5
- Optimizer : Adam(Adaptive Moment Estimation)
- Loss : Binary Crossentrophy

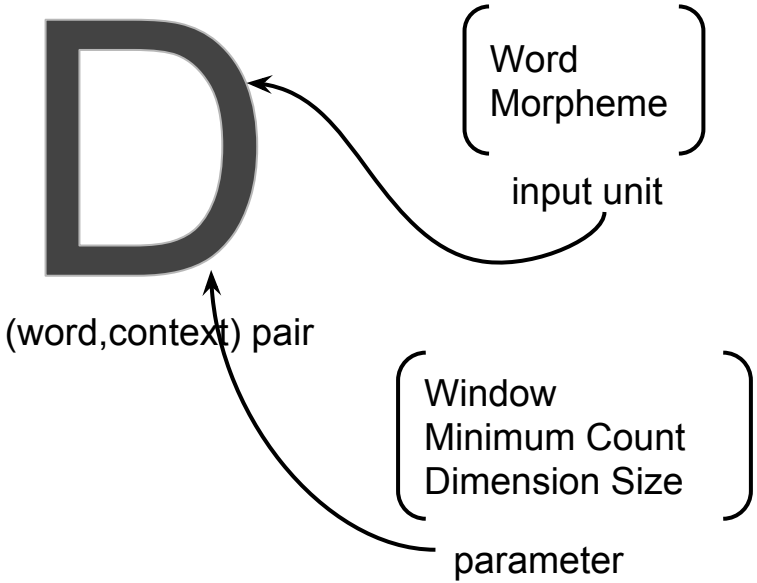
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50)	0
embedding_1 (Embedding)	(None, 50, 300)	88354800
lstm_1 (LSTM)	(None, 300)	721200
dense_1 (Dense)	(None, 2)	602

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

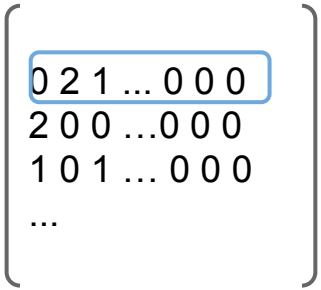
Optimizer: adam, Loss : binary_crossentropy, EarlyStopping : 5 (patience level)

How to Interpret Result ?

Src of information



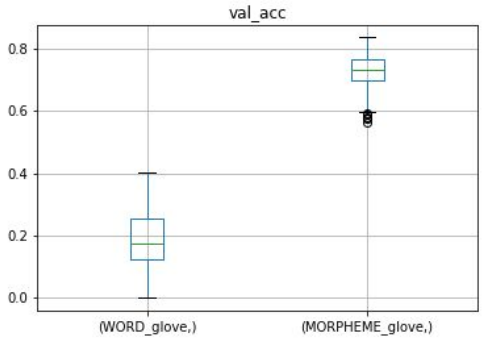
Co-occurence matrix



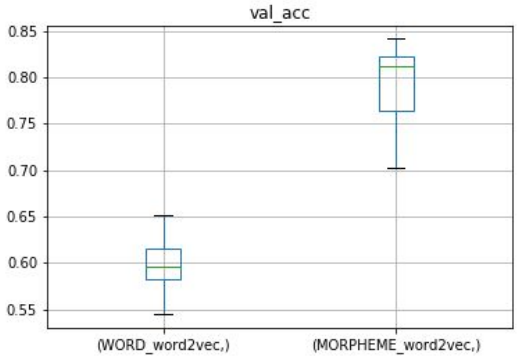
- # of vocabulary
- # of average freq
- # of average non zero columns

Result - Input Unit

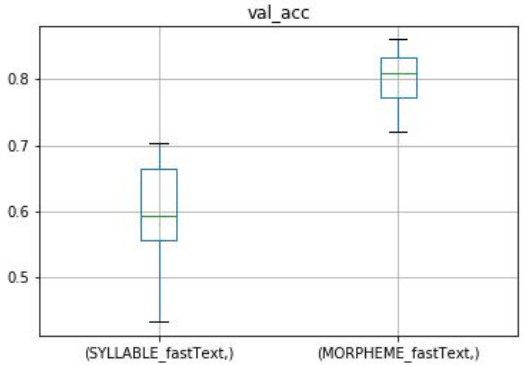
Glove



Skipgram



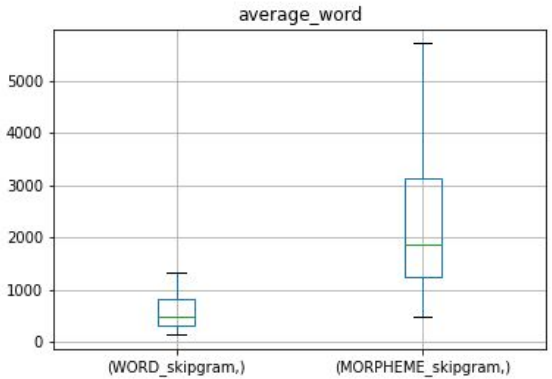
FastText_Syllable



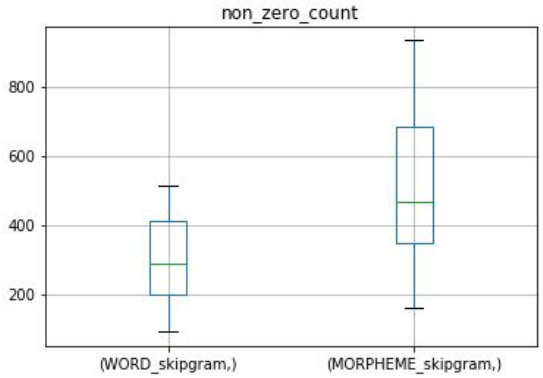
- 모든 모델에서 형태소 단위의 입력이 성능이 더 좋음

Result - Interpretation

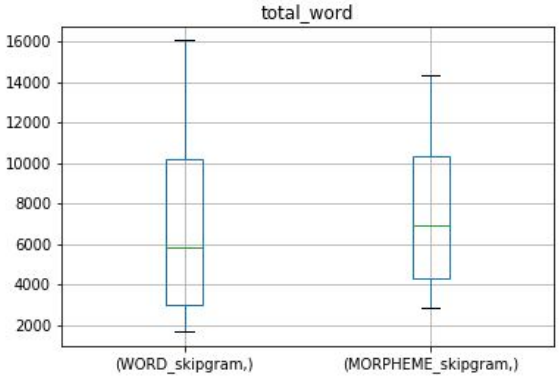
어휘당 평균 출현빈도



어휘당 문맥의 종류



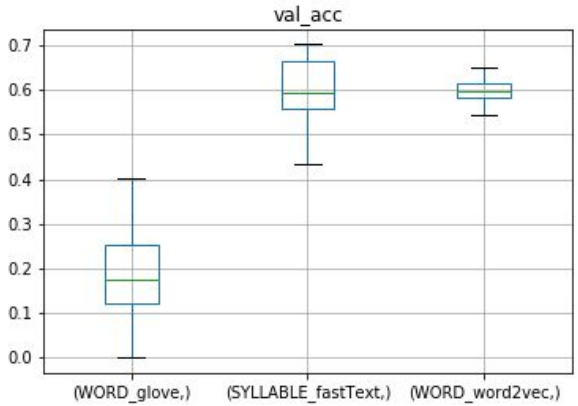
전체 단어 종류



- 전체 단어종류는 줄어들었지만,
- 어휘당 평균 출현빈도와 문맥의 종류는 증가했다

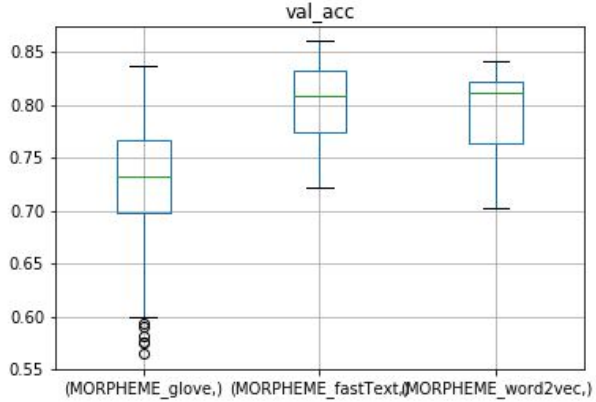
Result - between models

Word



- Glove < Skipgram < FastText

MORPHEME



- Glove < Skipgram < FastText

Result Interpretation

Tianze Shi and Ahiyuan Liu(2014)

- Bias term 간에 강한 연관
- SGNS가 최적화 하는 행렬과 Glove가 비슷

	Glove	Skipgram with Negative Sampling
Local cost function	$l_G(w_i, c_j) = f(X_{i,j}) \times \left(W_i \cdot C_j^T + b_{w_i} + b_{c_j} - \log X_{i,j}\right)$ $f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha}, & x < x_{max} \\ 1, & otherwise \end{cases}$ $X_{i,j} = \#(w_i, c_j)$ $\alpha = \frac{3}{4}, \quad x_{max} = 100$	$l_s(w_i, c_j) = X_{i,j} \times log\sigma(W_i \cdot C_j^T) + k \times \#(w_i) \cdot \frac{\#(c_j)}{\sum_w \#w} log\sigma(-(W_i \cdot C_j^T))$
Global cost function	$\sum_{i=1}^{\lvert V_W \rvert} \sum_{j=1}^{\lvert V_C \rvert} l_G(w_i, c_j)$	$\sum_{i=1}^{\lvert V_W \rvert} \sum_{j=1}^{\lvert V_C \rvert} l_s(w_i, c_j)$
$M = W_i \cdot C_j^T$	$W_i \cdot C_j^T = X_{i,j} - b_{w_i} - b_{c_j}$	$W_i \cdot C_j^T = \text{PMI}(w_i, c_j) - logk$ $= X_{i,j} - \log \#(w_i) - \log \#(c_j) + \log \sum_w \#(w) - logk$
Window Weighting Scheme	L = Maximum window size , d = disance from the focus word	
	$\frac{L - d + 1}{L}$	$\frac{L - \sqrt{d} + 1}{L}$

단어-문맥쌍 복원 비용

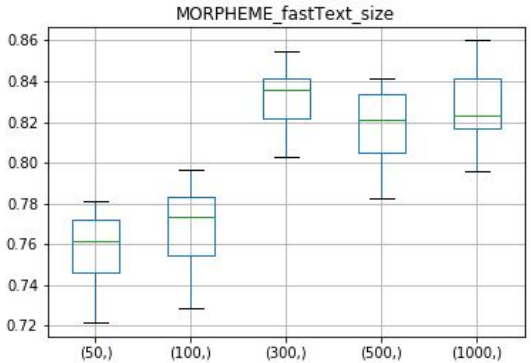
$l(w_i, c_j)$

어휘당 평균 출현 빈도가 확보되지 않았을때 더 큰 영향

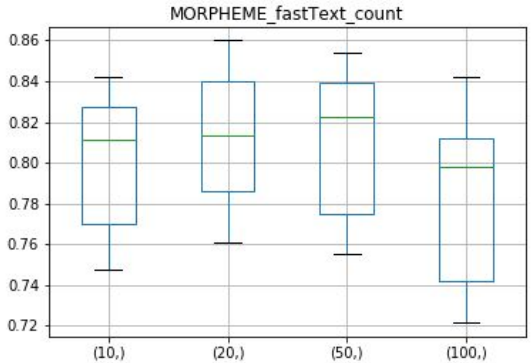
→두 모델간 단어 단위 성능 차이설명

Result - parameter

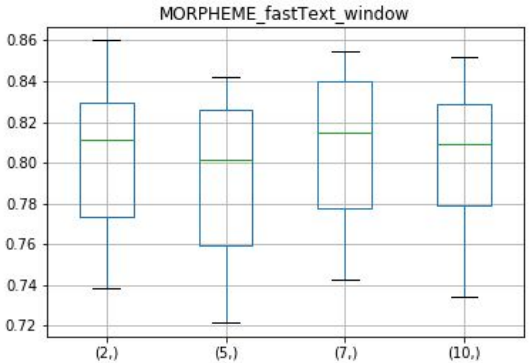
Dimension Size



Minimum Count

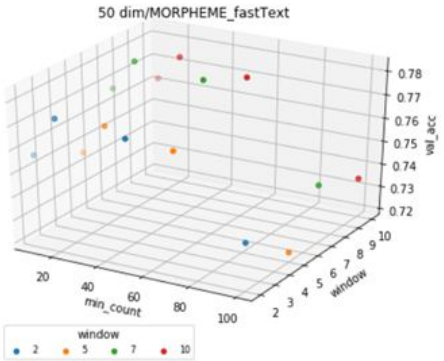


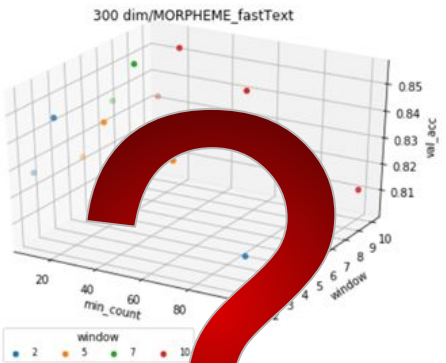
Window Size

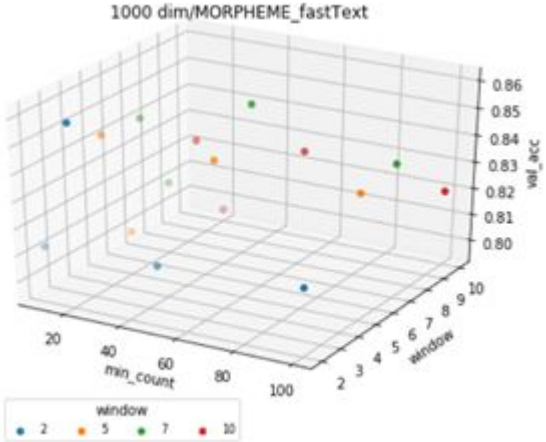


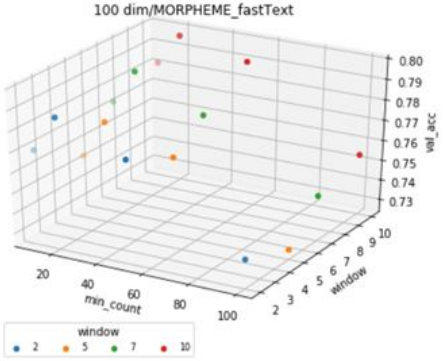
- size파라메터가 모델의 정확도에 가장 영향을 많이주었다
- window, minimum count 중 성능에 더 큰 영향을 주는 요소는?

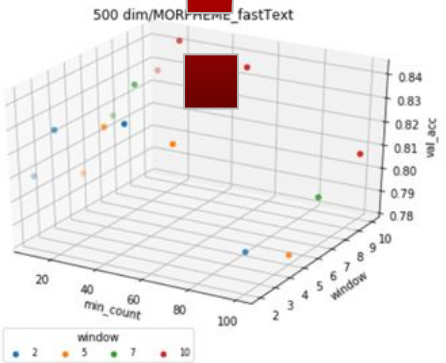
Window - Min Count Relation



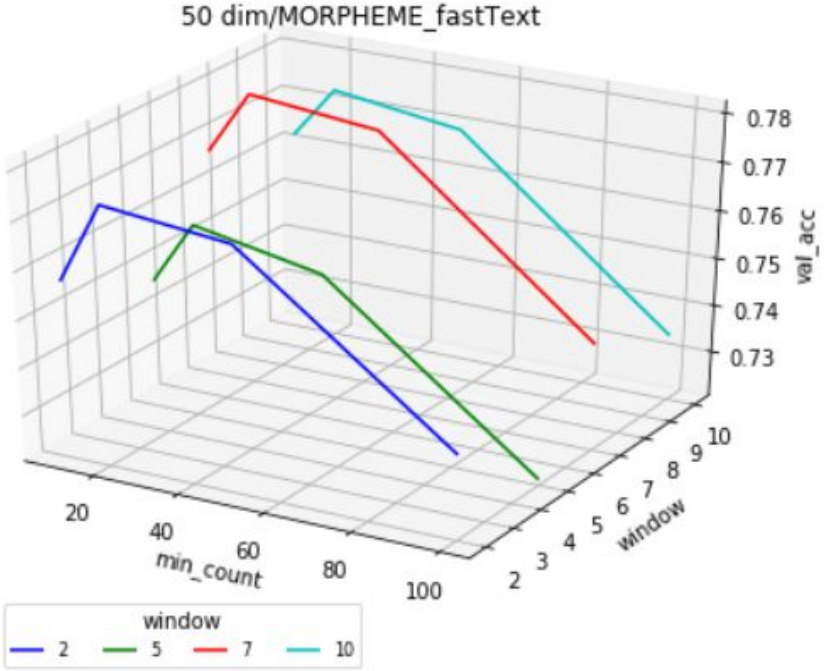








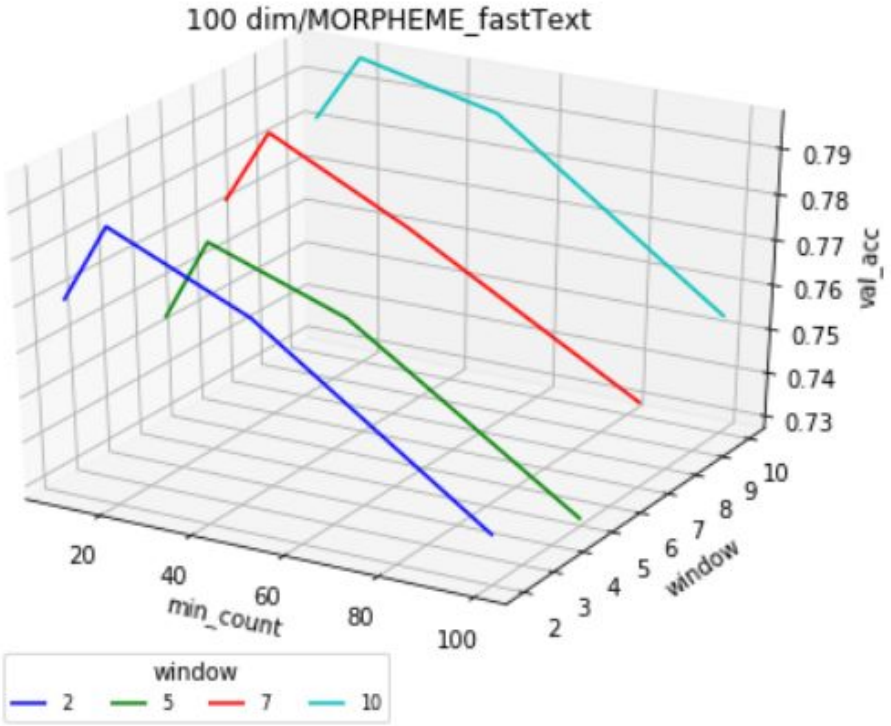
Result - Group By Window



Min 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]

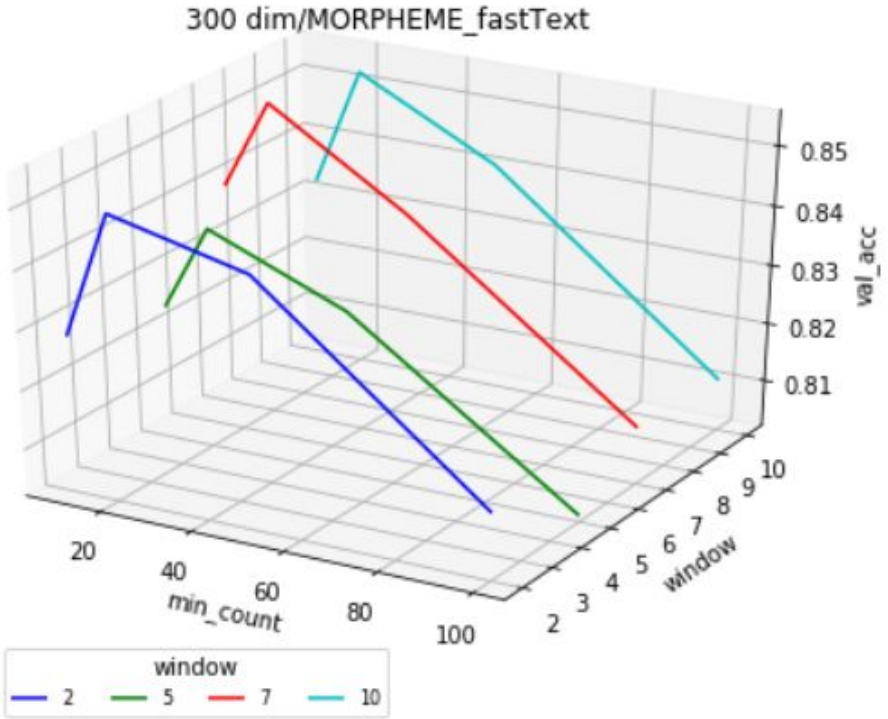
Result - Group By Window



Min 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]

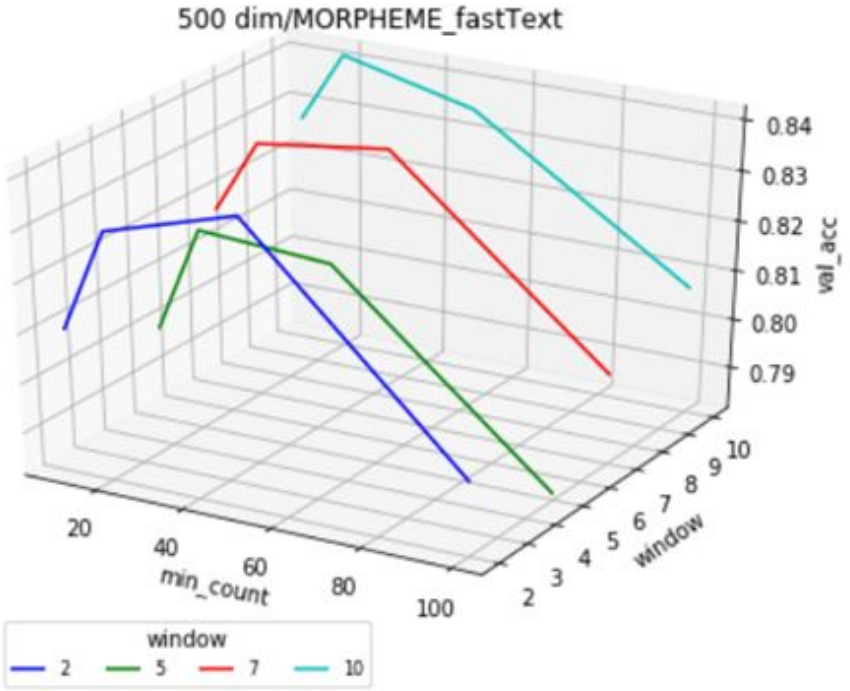
Result - Group By Window



Min 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]

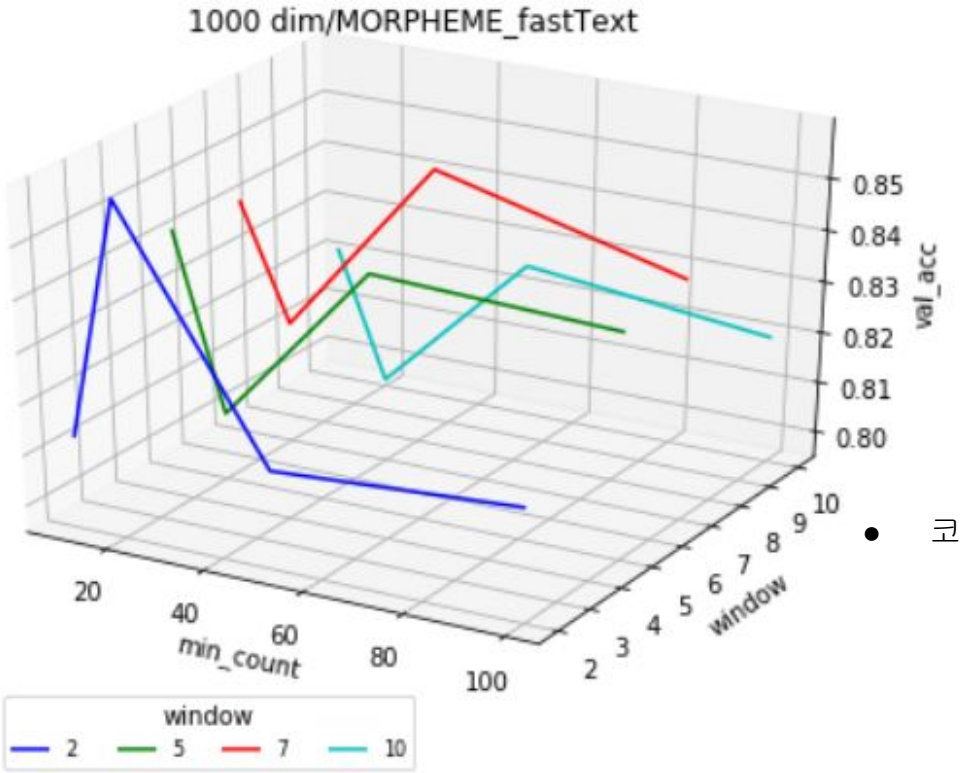
Result - Group By Window



Min 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]

Result - Group By Window

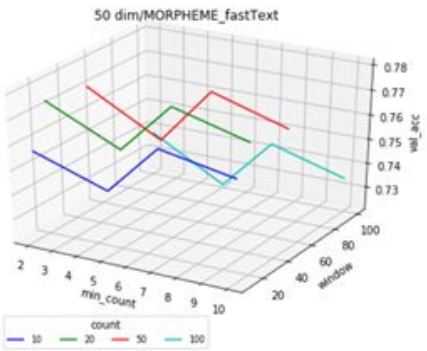


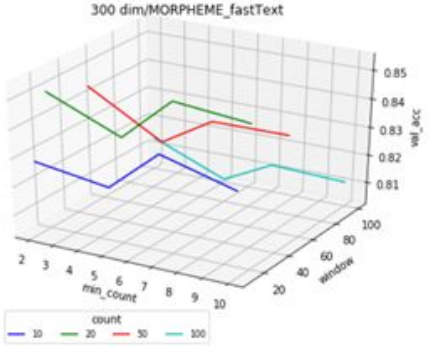
Min Count

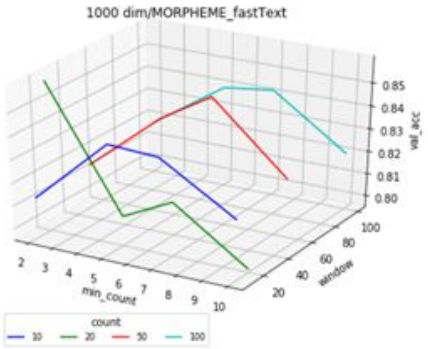
50 : [100 10 50 20]
100 : [100 10 50 20]
300 : [100 10 50 20]
500 : [100 10 50 20]
1000 : [100 10 50 20]

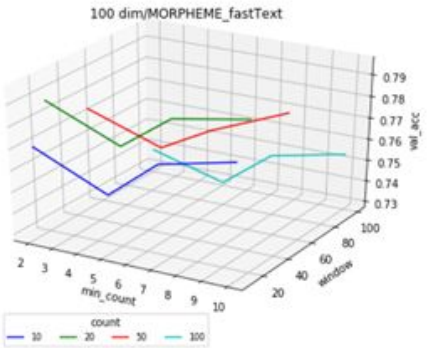
- 코퍼스마다 적정선의 최소출현빈도 파라메터가 존재

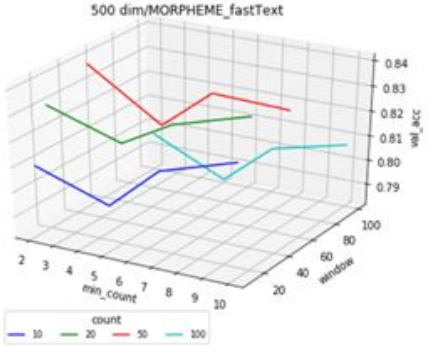
Result - Group By Min Count











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]

Result Interpretation

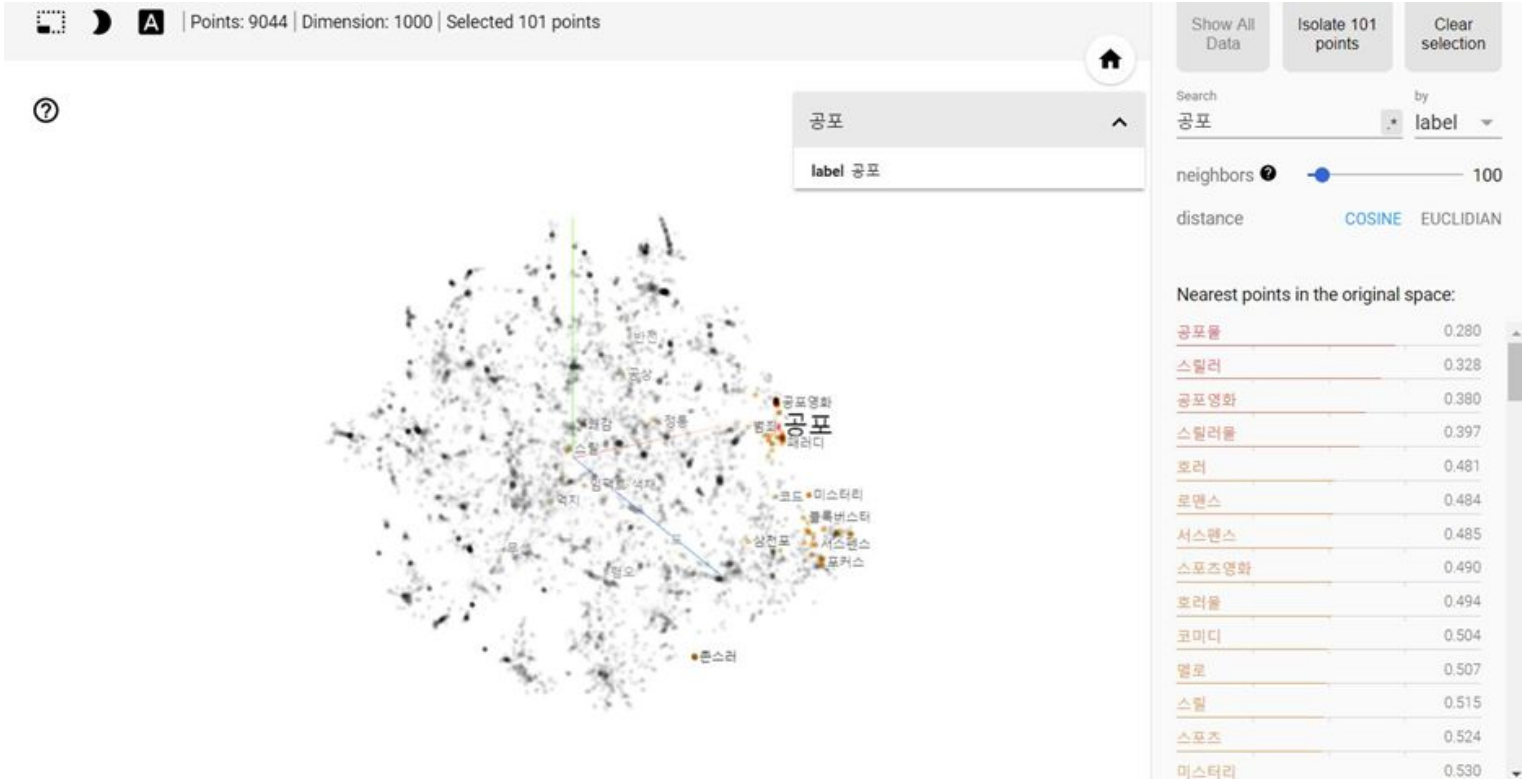
	FastText	Skipgram
출력층의 softmax 계산	$p(c\|w) = \frac{e^{s(c\|w)}}{\sum_{j=1}^{V_W} e^{s(c\|w)}}$	
s(c, w)	$s(c, w) = \sum_{g \in g_w} z_g^T C_c$	$s(c, w) = W_w * C_c$
Ex) 소프트웨어[7] , n=3	g_w 소프트웨어 = { < 소프, 소프트, 프트웨, 트웨어, 웨어 >, < 소프트웨어 >}	W^w 소프트웨어 = <소프트웨어>

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

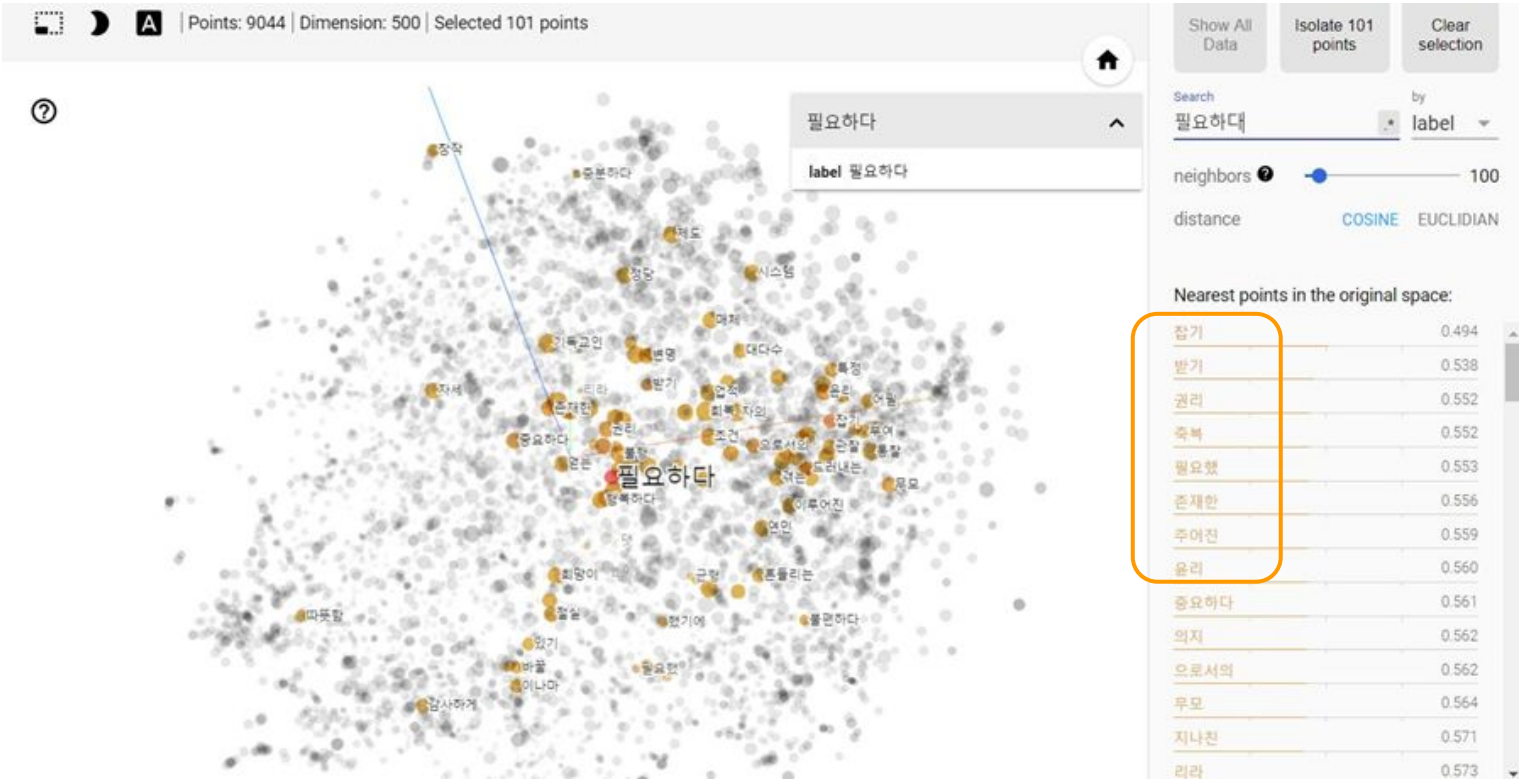
Result Interpretation - PCA

텐서보드의 Embedding Projector에 맵핑을 하였고,¶
PCA 보다는 TSNE의 클러스터가 두 모델간 차이를 더 잘 보여준다
*TSNE :
Perplexity = 25, iteration = 500

Result Interpretation - TSNE

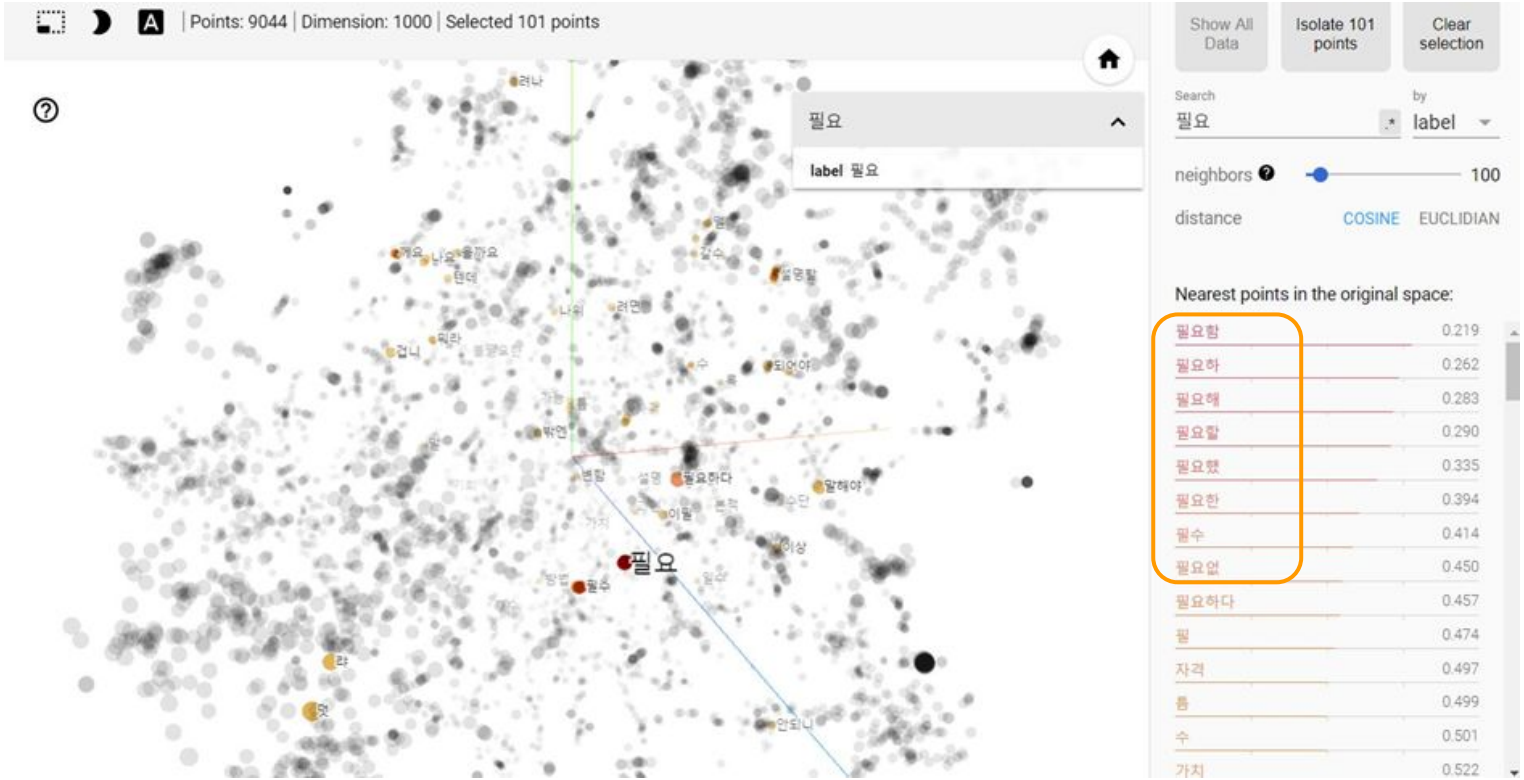


Skipgram - TSNE

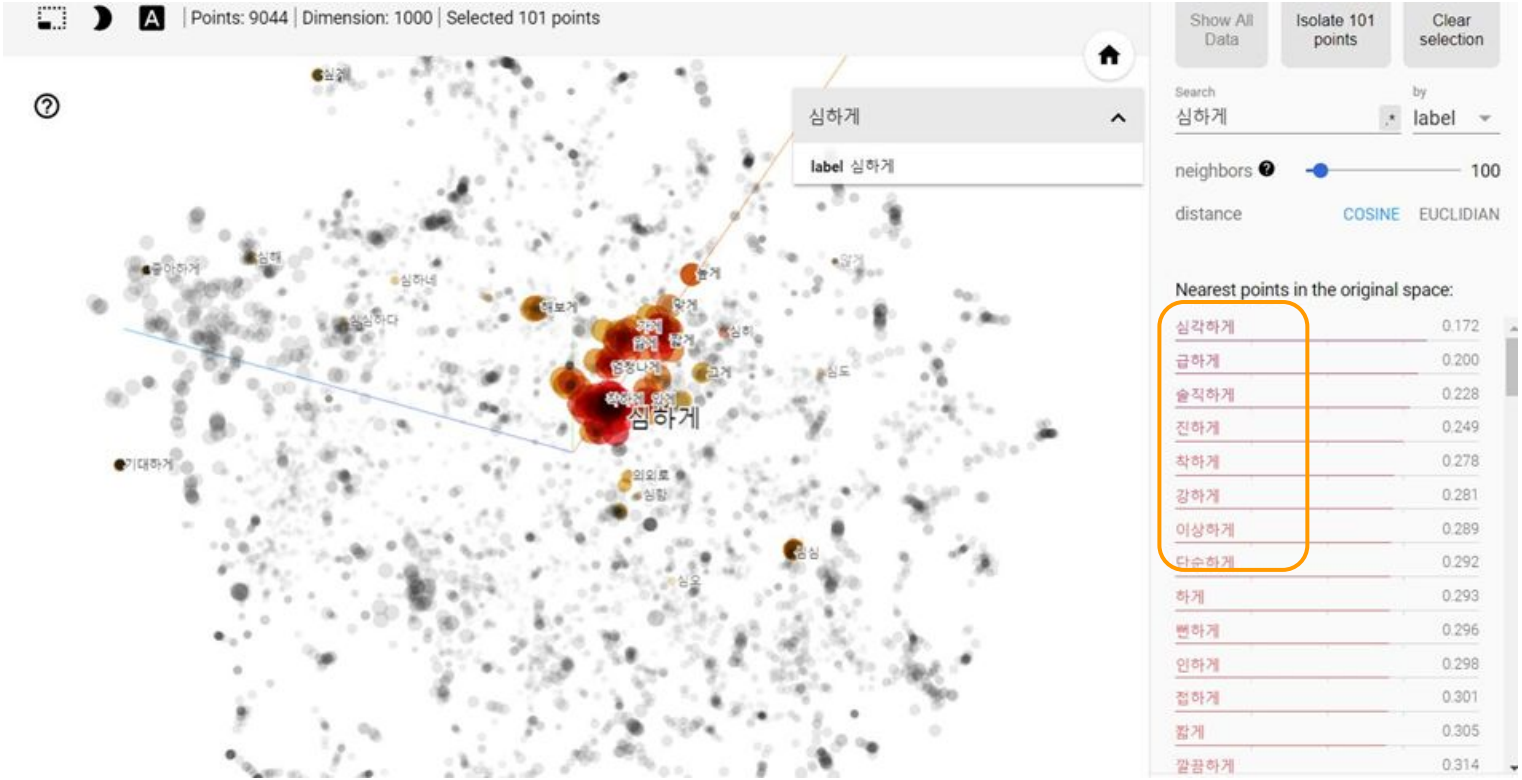


Skipgram - TSNE

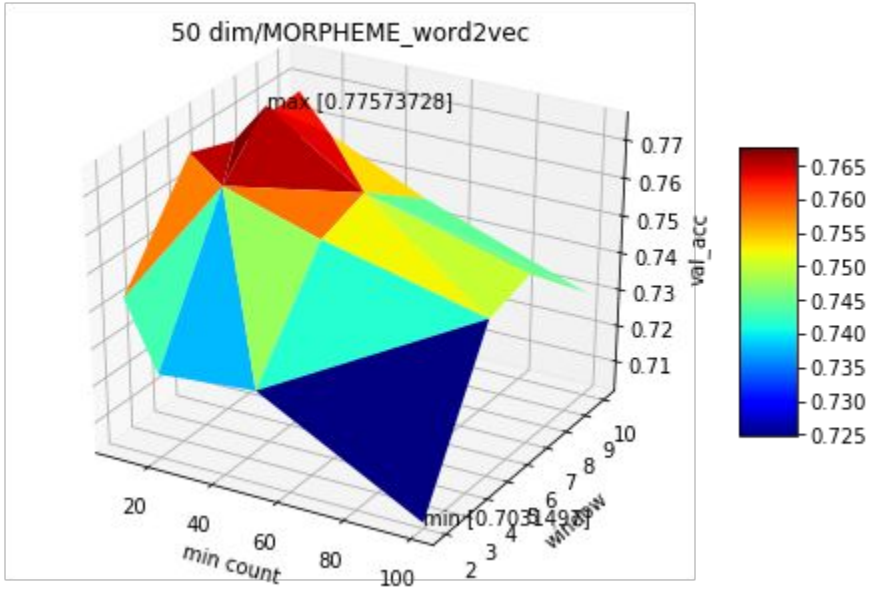
FastText - TSNE

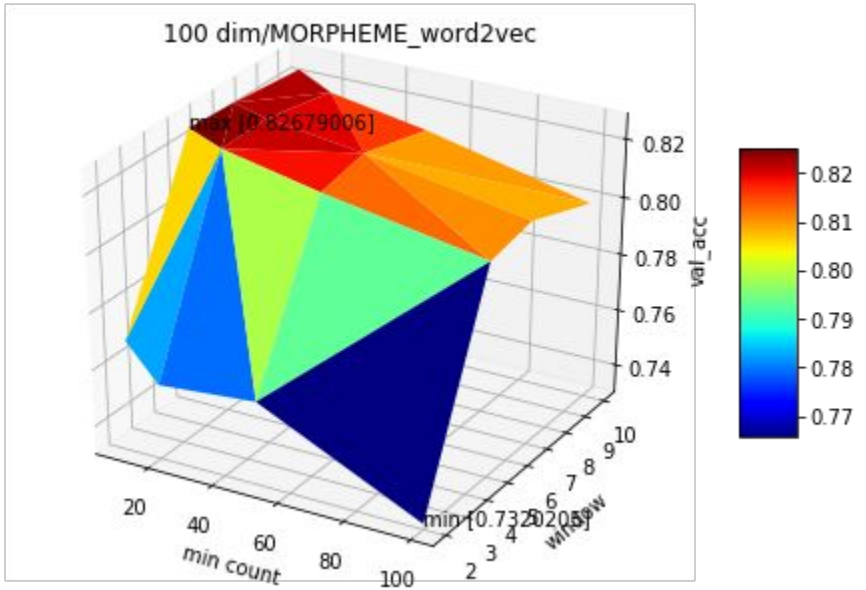


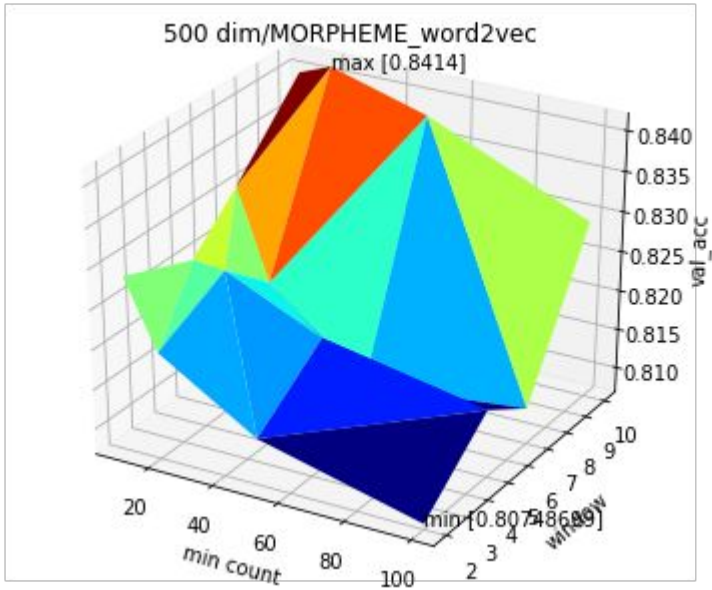
FastText - TSNE

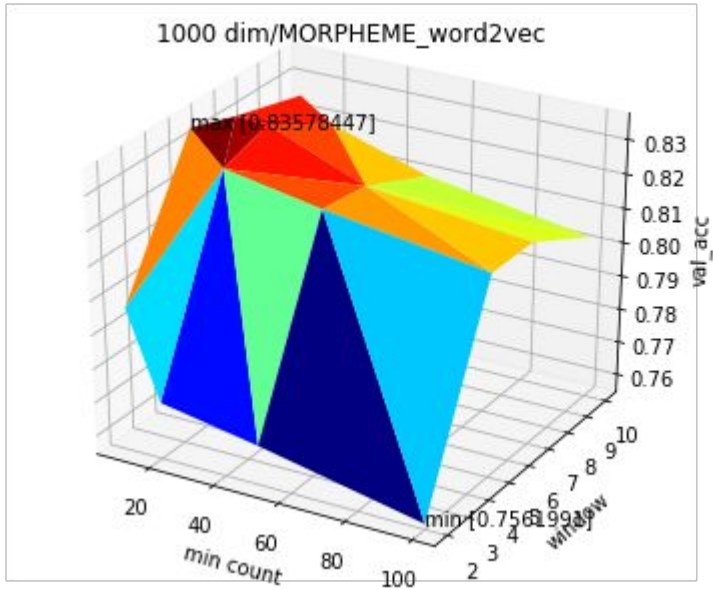


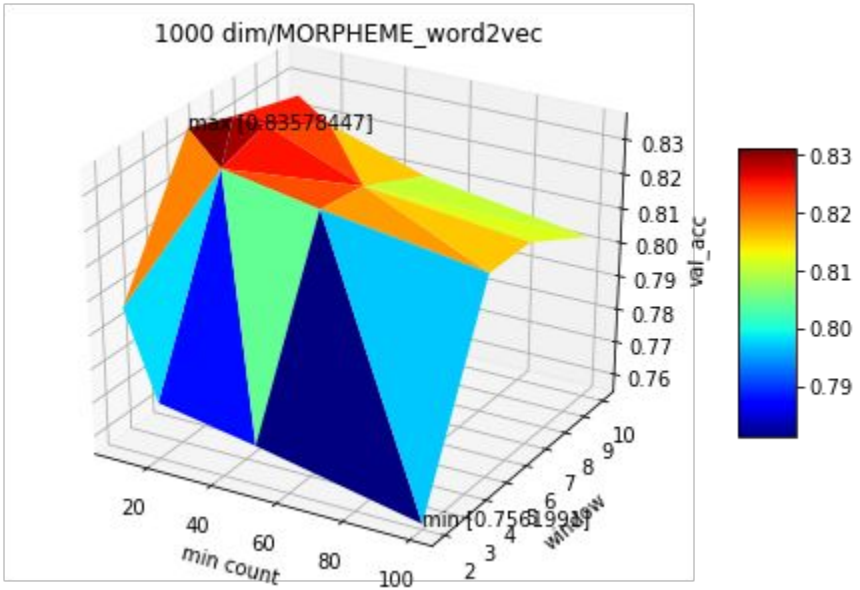
What about Skipgram?






500 dim/MORPHEME_word2vec
max [0.8414]
min [0.80748689]
val_acc
min count
window


1000 dim/MORPHEME_word2vec
max [0.83578447]
min [0.75619994]
val_acc
min count
window



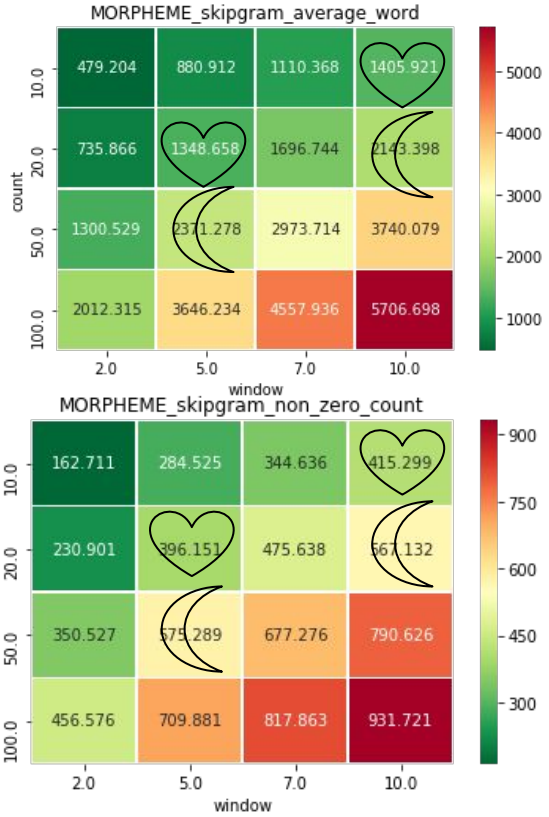
Window

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

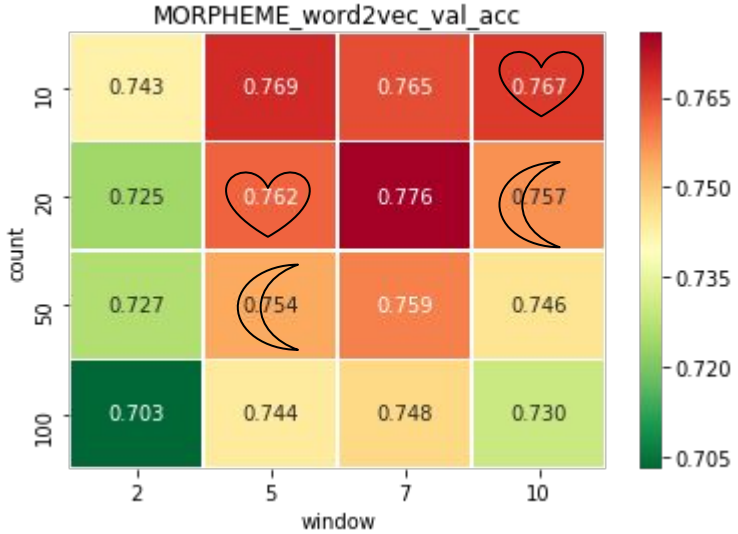
Min Count

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

Result - Interpretation

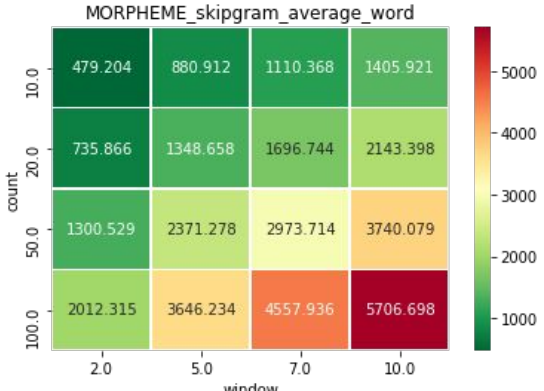


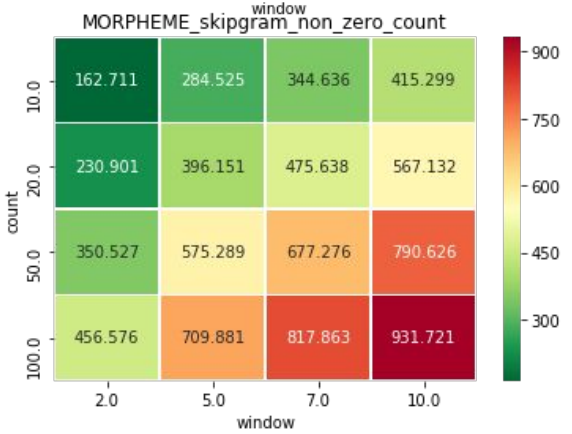
Dim = 50



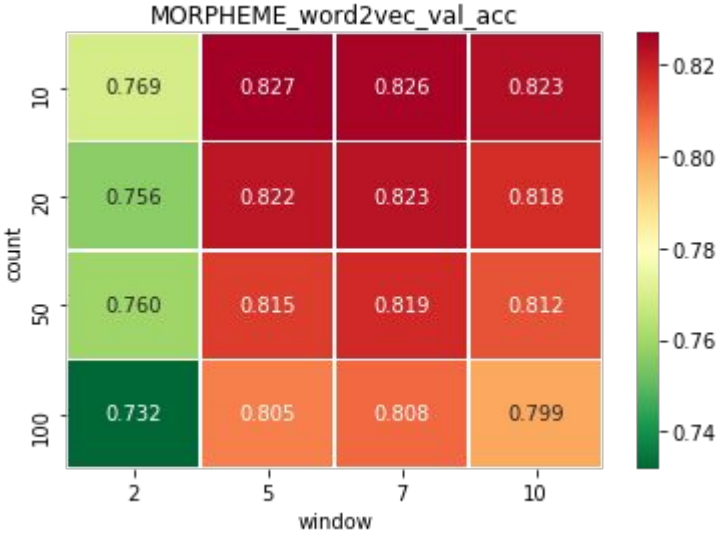
- 차원의 크기와 무관한게 일정한 지점에서 최고점을 형성
- f(average_word, non_zero_count) ≒ val_acc , ∃ f
  - Dimension size 마다 달라짐

Result - Interpretation

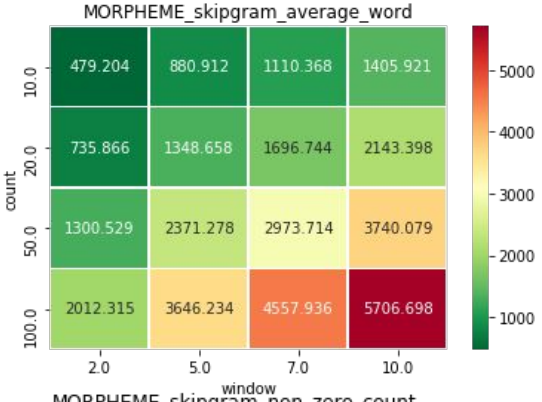


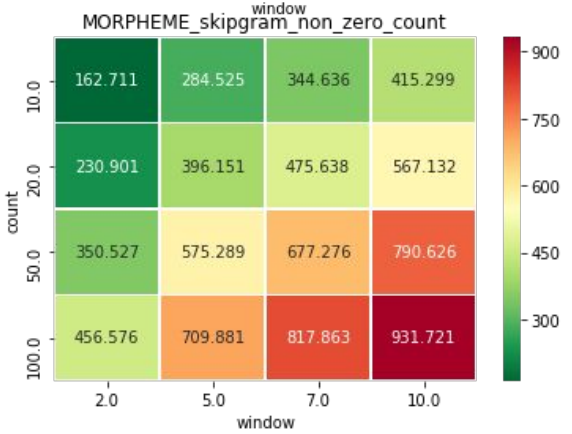


Dim = 100

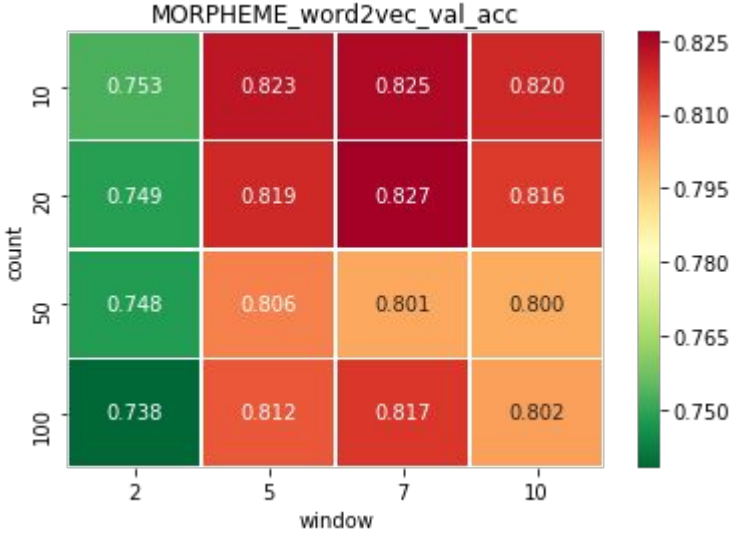


Result - Interpretation

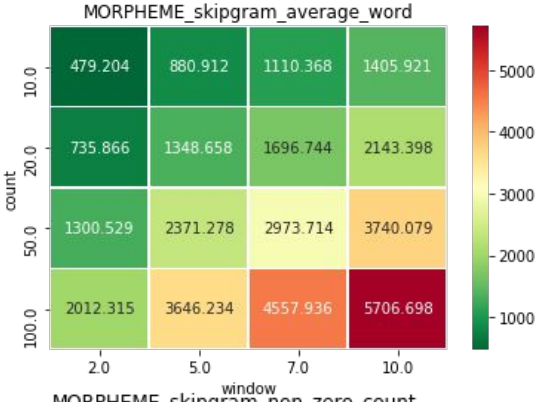


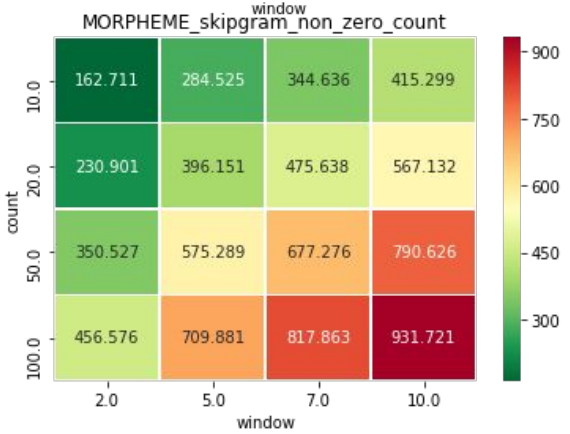


Dim = 300

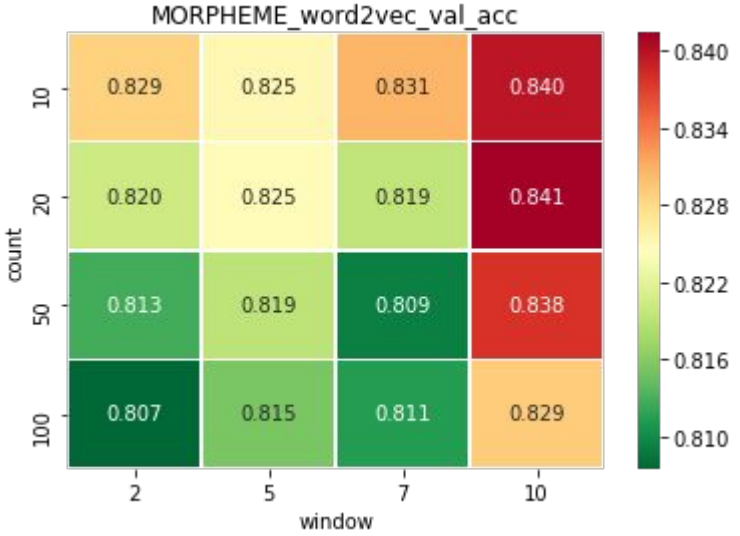


Result - Interpretation

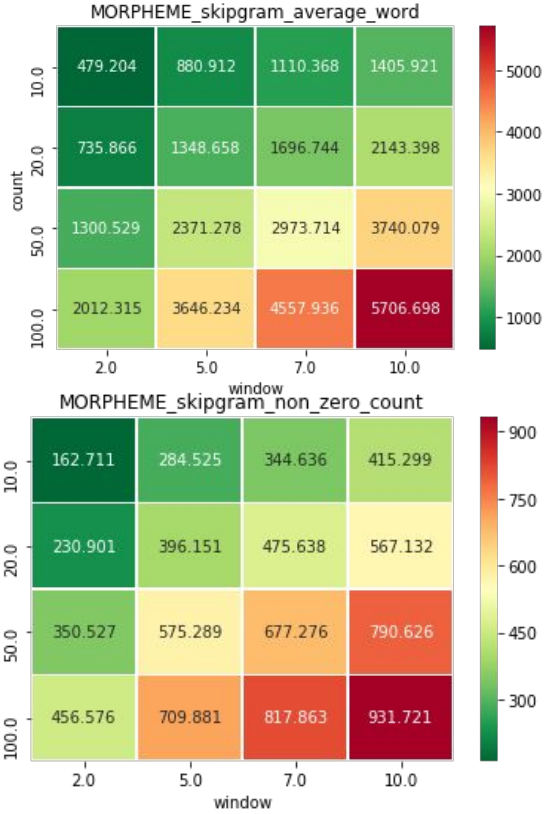




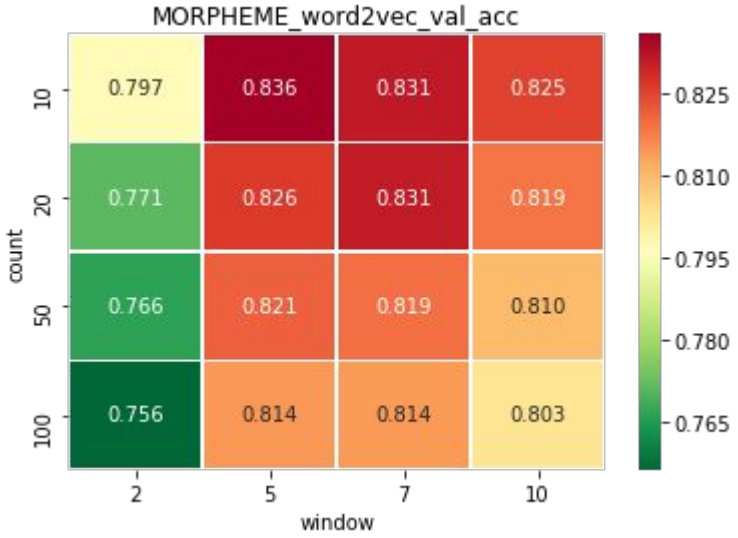
Dim = 500



Result - Interpretation

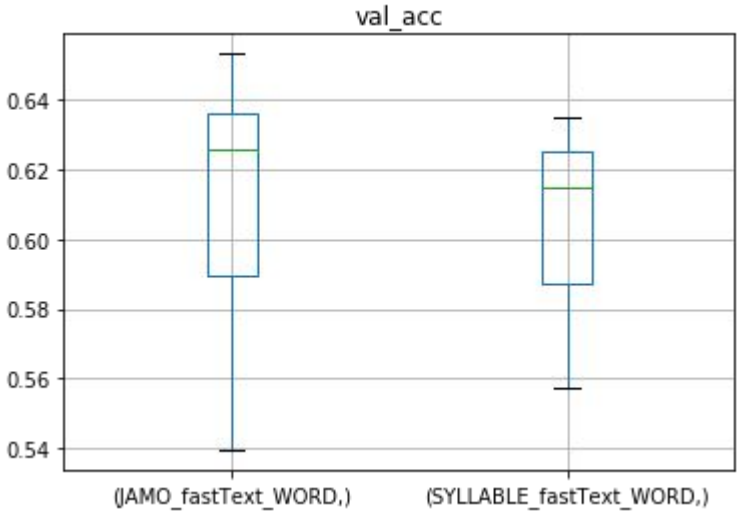


Dim = 1000



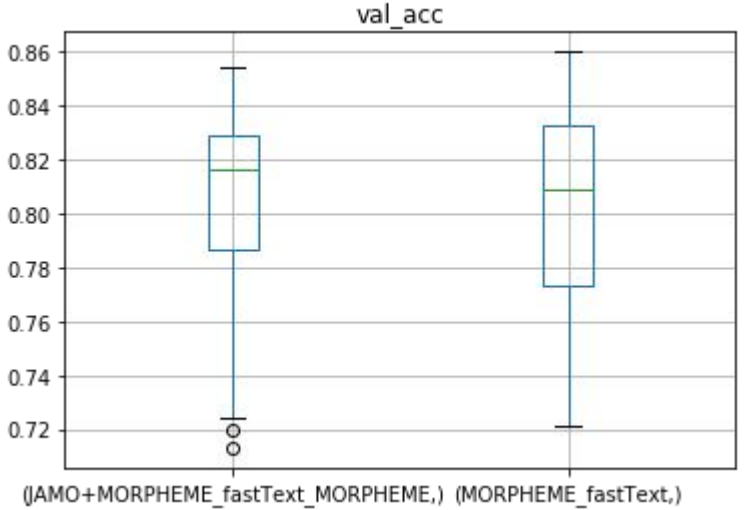
FastText : Grapheme(jamo) vs Syllable

Word



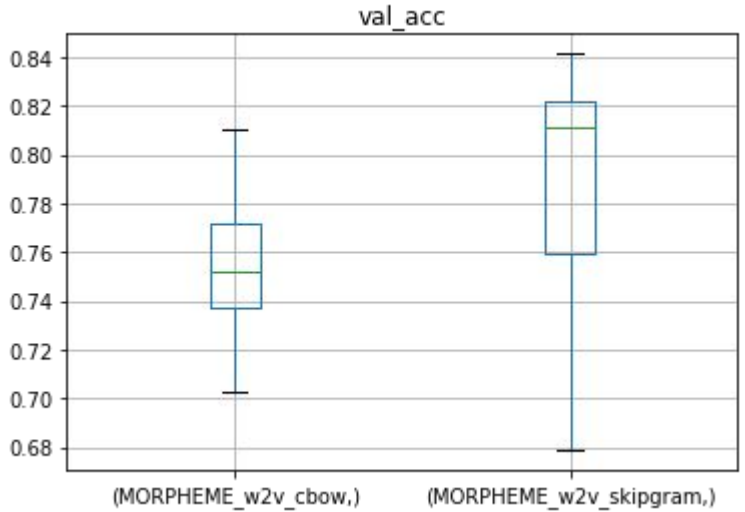
Grapheme > Syllable

Morpheme



Grapheme < Syllable

Word2vec : Skipgram vs Cbow



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

Conclusion

Input Unit

Morpheme 단위의 입력 사용

Parameter

Dimension > Min Count > Window 순으로 최적화

Model

FastText > Skipgram ~ Glove

단, Glove의 경우 어휘당 출현 빈도에 민감

FastText의 경우 자모 단위의 입력이 노이즈로 작용할 수도 있음

Word2vec : Skipgram > Cbow

Reference

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.

...

Thank You !, Q&A