Hyperparameter Tuning in Embedding

0. Reference

- Improving Distributional Similarity with Lessons Learned from Word Embeddings, Levy et al.

1. Intro

이 논문은 새로운 임베딩 기법을 제시하기 보다는, 기존의 임베딩 기법의 hyperparameter을 바꾸거나 방법을 약간 변형해서 결과를 비교해본다. 어떻게 보면, 임베딩 벡터의 품질을 향상시키거나 떨어뜨리는 방법을 각 임베딩 기법별로 제시하여 실제 임베딩 벡터를 사용하는 practical tips을 얻을 수 있다.
이 논문에서 말하는 hyperparameter는 꽤 넓은 범위를 의미한다. 예를 들어, word2vec에서 negative-sampling의 수, context windows의 종류 등을 모두 hyperparameters라고 칭한다.
이 논문에서는 이러한 hyperparameters를 분명하게 짚고 넘어가고 전통적인 count-based 접근법에 어떤 식으로 적용할 수 있는지 살펴본다. 예를 들어, word2vec에서 negative sampling을 할 때, smoothing을 하는데, 이 방법이 PPMI-based 방법에 적용하여 성능 향상을 야기한다. 논문에서 비교하고자 하는 임베딩 기법은 아래와 같다.

- count-based distributional models
  - Pointwise Mutual Information
  - Singular Value Decomposition
- neural network-based models (prediction-based)
  - word2vec
  - Glove

어떤 논문에서는 count-based 임베딩 벡터가 neural network 기반 임베딩 벡터보다 성능이 좋지 않다고 말하는데 이러한 주장에 대해 이 논문은 hyperparameter만 잘 조절한다면 두 기법의 임베딩 벡터 품질은 크게 차이가 없다고 반박한다.

Notation

- collection of words $w \in V_W$. V_W 는 word vocabularies
- context words $c \in V_C$. V_C 는 context vocabulries

- collection of observed word-context pairs as D
- $\#(w, c)$ to denote the number of times the pair (w, c) appears in D
- $\#(w) = \sum_{c' \in V_C} \#(w, c^{'})$, $\#(c) = \sum_{w' \in V_W} \#(w^{'}, c)$
- 각 단어 w는 $\vec{w} \in \mathbb{R}^d$ 벡터와 관련된다.
- 마찬가지로 각 context word c는 $\vec{c} \in \mathbb{R}^d$ 벡터와 관련된다.
- 방법 x에 의해서 만들어진 임베딩 벡터는 W^x로 표기한다. (W^{SGNS}, C^{SVD})
- dot product가 cosine similarity와 동일한 값을 가지도록 모든 벡터를 정규화한다.
- 크기가 L인 window를 단어 w_i에 적용할 때 context words는 아래와 같다.

$$w_{i-L}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+L}$$

2. Transferable Hyperparameters

이제 word2vec, glove의 hyperparameters를 count-based method에 어떻게 적용하는지 알아본다. 이를 아래의 세 가지 방법으로 나눈다.

- pre-processing hyperparameters, which affect the algorithm's input data
- association metric hyperparameters, which define how word-context iteractions are calculated
- post-preprocessing hyperparameters, which moify the resulting word vectors

아래 표는 hyperparameter별로 어떤 값을 사용했고, 이를 적용한 method를 정리한 표이다.

Hyper-parameter	Explored Values	Applicable Methods
`win`	2, 5, 10	All
`dyn`	none, with	All
`sub`	none, dirty, clean†	All
`del`	none, with†	All
`neg`	1, 5, 15	PPMI, SVD, SGNS
`cds`	1, 0.75	PPMI, SVD, SGNS
`w+c`	only w, $w + c$	SVD, SGNS, GloVe
`eig`	0, 0.5, 1	SVD
`nrm`	none†, row, col†, both†	All

예를 들어, hyperparameter에서 win는 어떤 크기의 windows를 사용할지에 대한 것이며, 이 실험에서는 2, 5, 10 크기의 windows를 사용했고 모든 방법, 즉 SGNS, Glove, PPMI, SVD에 적용했다는 뜻이다. 이제 각 hyperparameter별 내용을 살펴보겠다.

2.1 pre-processing hyperparameters

- Dynamic Context Window (dyn)
  window를 이용하여 context word을 뽑을 때, 각 context word에 어떤 가중치를 부여할지에 대한 내용이다. 중심 단어와 가까운 context 단어는 더 높은 가중치를 부여 받는다. glove와 word2vec은 이러한 weighting 전략을 사용한다. 예를 들어 word2ec은 문맥 단어와 중심 단어까지의 거리 / window size 값을 가중치를 준다. 즉, 크기가 5인 window에 대해서는 가중치가 $\frac{5}{5}, \frac{4}{5}, \frac{3}{5}, \frac{2}{5}, \frac{1}{5}$ 이다. (근데 이러면 가까운 단어가 더 적은 가중치가 아닌가?...)
  dynamic이라 불리는 이유는 word2vec이 각 token의 window 크기에 대해 1부터 L까지 고르게 뽑기 때문이다. 실험에서는 모든 method에 대해 word2vec의 dyn을 적용하였다.
  (SVD, PPMI에 대해서 dyn을 어떻게 적용하는지는 자세하게 나오지 않았다. 생각해볼 숙제)

- Subsampling (sub)
  자주 나오는 단어를 dilute하는 것이다. 한 방법은 corpus가 word-context pairs로 만들어지기 이전에 tokens을 제거하는 것이다. 이는 많은 token에 대해 context window size를 증가시키는데, 제거 이전에 크기가 L인 window를 통해 서로 만날 수 없었던 단어들이 제거 이후 만나기 때문이다. 이러한 subsampling 기법을 'dirty'라고 부른다.
  반면에, context window size에 영향을 주지 않고 subsampled 단어를 없애는 기법을 'clean' subsampling이라고 부른다.

- Deleting Rare Words (del)
  training corpus에서 희귀한 단어들을 없애는 방법이다. 이 방법은 결과에 큰 영향을 끼치지 않는 것으로 나와, 본문에서 따로 언급하지 않는다.

2.2 Association Metric Hyperparameters
여기서는 SGNS로부터 착안하여, PMI의 두 변형을 살펴본다.

- Shifted PMI (neg)
  SGNS는 negative samples, k라는 hyperparameter가 있다. Levy에 따르면 SGNS의 word context matrix는 (w, c) : $PMI(w, c) - logk$를 optimize 한다. k에 의해서 PMI가 shift된 형태인데, 이를 PPMI에 적용해볼 수 있다.

$$SPPMI(w, c) = max(PMI(w, c) - logk,\ 0)$$

  본 논문의 실험에서는 $k = 1, 5, 15$ 값으로 설정한다.

- Context Distribution Smoothing (cds)
  word2vec에서는 negative examples가 smoothed unigram distribution을 통해 추출된다. 이러한 smooting 개념을 PMI에도 적용할 수 있다.

$$PMI_\alpha(w,c) = log\ \frac{\hat{P}(w,c)}{\hat{P}(w)\hat{P}_\alpha(c)}$$

$$\hat{P}_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha}$$

  이러한 smooting은 희귀한 단어에 대한 PMI의 bias를 어느 정도 보정해준다. 이는 $\hat{P}_\alpha(c) > \hat{P}(c)$ 임을 통해 알 수 있다. 이 방법은 실험에서 매우 효과적임이 밝혀졌다. 여기서는 $\alpha = 1$(unsmoothed), $\alpha = 0.75$(smoothed) 두 값을 적용한다.

2.3 Post-processing Hyperparameters

여기서는 알고리즘의 결과물, word vectors을 변경하는 hyperparameters를 제시한다.

- Adding Context Vectors (w+c)
  Pennington은 Glove의 결과물에 word vector에 context vector를 더해서 쓸 것을 제안하였다. 예를 들어, 고양이 단어 벡터를 아래와 같이 표현하는 것이다.

$$v_{cat} = w_{cat} + c_{cat}$$

  원래 이 개념은 ensemble 방법에서 착안하였다. 본 논문에서는 cosine similarity 관점으로 접근하는데, 자세한 내용은 생략한다.

- Eigenvalue Weigting (eig)
  SVD를 이용한 word vector와 context vector는 아래와 같이 정의된다.

$$W^{SVD} = U_d \cdot \Sigma_d,\ C^{SVD} = V_d$$

  SVD의 정의에 의해서 U_d, V_d는 orthonormal하다. 그렇지만 $U_d \cdot \Sigma_d$는 그렇지 않다. 즉, W^{SVD}, C^{SVD}가 symmetric하지 않다는 것이다. word2vec에서는 두 행렬이 orthonormal하여 symmetric하다. symmetry는 아래와 같이 충족될 수 있다.

$$W = U_d \cdot \sqrt{\Sigma_d},\ C = V_d \cdot \sqrt{\Sigma_d}$$

  또는 eigenvalue matrix을 아예 제외할 수도 있다.

$$W = U_d,\ C = V_d$$

  이러한 symmetry가 왜 performance 향상에 도움을 주는지 이론적으로 밝혀지지는 않았다. 본

논문에서는 위에서 정의한 symmetric matrix와 원래의 SVD 정의에 따른 행렬 분해를 실험해 본다.

- Vector Normalization (nrm)
  행, 열, 행 열 모두 정규화하는 방법이있다. 실험에서는 W의 행을 정규화하는 것이 성능이 지속적으로 높게 나왔다.

3. Results Summary

- count-based 방법이 prediction-based 방법보다 더 좋을까?
  논문의 실험 결과를 보면 그렇다고 볼 수 없다. 오히려, SVD나 SPPMI가 지속적으로 더 좋은 성능을 보이는 보이는 task가 여럿 있었다. 여러 논문에서 prediction-based 방법이 count-based 방법보다 더 좋다고 주장하는데, 이는 hyperparameter를 조정하지 않아서 비롯된 결과이다. 'vanilla' PPMI나 SVD는 word2vec이나 glove에 비해 성능이 좋지 않지만, $cds = 0.75$인 PPMI나, $eig = 0, 0.5$인 SVD는 word2vec, glove보다 여러 task에서 높은 정확도를 보였다.

- glove가 SGNS보다 좋을까?
  glove 논문 저자 Pennington은 glove가 더 좋다고 말하지만, hyperparameter를 조정한다면 꼭 그렇지도 않다. $w + c$ 벡터를 사용한다든가, 더 다양한 analogy task를 수행한다면 결과가 다르게 나온다.

- 3CosMul가 3CosAdd보다 더 많은 analogies를 나타낼까?
  Levy는 3CosMul의 결과가 3CosAdd의 결과보다 전반적으로 더 높은 정확도를 보임을 밝혔고 이는 본 논문의 결과와 일치한다.

4. Practical Recommendations

- PPMI에 대해서는 항상 smooting ($cds = 0.75$)를 사용하자. 이 조정을 통해 prediction-based 방법보다 더 좋은 정확도를 얻을 수 있다.

- SVS를 $eig = 1$로 사용하지 말자. 이보다는 symmetric 변형인 $eig = 0, 0.5$를 사용하자.

- SGNS는 robust baseline이다. 모든 task에서 가장 좋은 정확도를 내는 것은 아니지만, robust baseline이다. 즉, 다른 모델보다 확연하게 정확도가 낮은 것은 아니다. 게다가 SGNS는 빠르게 훈련할 수 있고 disk space 등도 적게 사용이 된다.

- SGNS를 사용할 때, 많은 negative samples을 사용하자.

- SGNS나 glove를 사용할 때는, 추가적인 훈련이 필요하지 않으므로 $w + c$를 한번 시도해 보는 것이 좋다.