Word Vectors and Word Senses

2020年6月26日　　11:21

##00_背景

在NLP的任务中，尤其是基于神经网络的的语言模型，将词表示为一个可以输入到模型中是非常重要的一个步骤。并且词表示的好坏和性能决定了是否能将一些关键信息作为语言模型的输入，对模型的输出的质量是非常重要的。

##01_相关工作

- skip-gram
- 基于SVD的LSA方法，利用了全局特征的矩阵分析
- Word2vec，利用局部上下文

##02_模型

- **GloVe模型**
  - 对所有无监督学习方法(有哪些方法？？ SVD， skip-gram)学习词表示来说，语料库词的统计是主要的信息源
  - 目前方法仍有两个问题
    - 如何从统计中生成意义
    - 词向量如何表示意义
  - Utilizing this main benefift of count data while simutatienously capturing the meaningful linear substructrues prevalent in recent log-bilinear prediction-based method like word2vec. GloVe is a new global log-bilinear regression model for the unsupervised learning of word reprensentations that outperforms other models on word analogy, word similarity and named entity recogonition tasks

- **数学描述**

Co-occurrence Matrix

$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$

- X：word-word co-occurrence counts matrix，X_{ij} tabulate the number of times word j occurs in the context of word *i*;
- $X_i = \sum_k X_{ik}$：the number of times any word appears in the context of word *i*;
- P_{ij}：the probability that word j appear in the context of word *i*.

$F(w_i, w_j, \widetilde{w_i}) = \frac{P_{ik}}{P_{jk}}$

- $w \in \mathbb{R}^d$：word vectors
- $\tilde{w} \in \mathbb{R}^d$: separate context word vectors

$F\left((w_i - w_j)^T \widetilde{w_k}\right) = \frac{F(w_i^T \widetilde{w_k})}{F(w_j^T \widetilde{w_k})}$

- $F(w_i^T \widetilde{w_k}) = P_{ik} = \frac{X_{ik}}{X_i}$
- $w_i^T \widetilde{w_k} = log(P_{ik}) = log(X_{ik}) - log(X_i)$
- $w_i^T \widetilde{w_k} + b_i + \widetilde{b_k} = log(X_{ik})$
- $J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \widetilde{w_j} + b_i + \widetilde{b_j} - logX_{ij})^2$
- $f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^a & if\ x < x_{max} \\ 1, otherwise \end{cases}$

- 遗留问题
  a. 这个模型如何解决上述的两个问题?
  b. 意义指的是什么？如何在词向量中表示意义?

##03_实验

- GloVe
- 基于Pytorch的GloVe

```
import torch
import torchtext.vocab as vocab

print([key for key in vocab.pretrained_aliases.keys() if "glove" in key])
cache_dir = "/home/kesci/input/GloVe6B5429"
glove = vocab.GloVe(name='6B', dim=50, cache=cache_dir)
print("一共包含%d个词。" % len(glove.stoi))
print(glove.stoi['beautiful'], glove.itos[3366])
```

- 由于词向量空间中的余弦相似性可以衡量词语含义的相似性，可以通过寻找空间中的 k 近邻，来查询单词的近义词。

```
def knn(W, x, k):
    '''
    @params:
        W: 所有向量的集合
        x: 给定向量
        k: 查询的数量
    @outputs:
        topk: 余弦相似性最大k个的下标
        [...]: 余弦相似度
    '''
    cos = torch.matmul(W, x.view((-1,))) / (
        (torch.sum(W * W, dim=1) + 1e-9).sqrt() * torch.sum(x * x).sqrt())
    _, topk = torch.topk(cos, k=k)
    topk = topk.cpu().numpy()
    return topk, [cos[i].item() for i in topk]

def get_similar_tokens(query_token, k, embed):
    '''
    @params:
        query_token: 给定的单词
        k: 所需近义词的个数
        embed: 预训练词向量
    '''
    topk, cos = knn(embed.vectors,
                    embed.vectors[embed.stoi[query_token]], k+1)
    for i, c in zip(topk[1:], cos[1:]):  # 除去输入词
        print('cosine sim=%.3f: %s' % (c, (embed.itos[i])))

get_similar_tokens('chip', 3, glove)
# cosine sim=0.856: chips
# cosine sim=0.749: intel
# cosine sim=0.749: electronics

get_similar_tokens('baby', 3, glove)
# cosine sim=0.839: babies
# cosine sim=0.800: boy
# cosine sim=0.792: girl

get_similar_tokens('beautiful', 3, glove)
# cosine sim=0.921: lovely
# cosine sim=0.893: gorgeous
# cosine sim=0.830: wonderful
```

- 除了求近义词以外，还可以使用预训练词向量求词与词之间的类比关系，例如“man”之于“woman”相当于“son”之于“daughter”。求类比词问题可以定义为：对于类比关系中的4个词“a 之于 b 相当于 c 之于 d”，给定前3个词 a,b,c 求 d。求类比词的思路是，搜索与 vec(c)+vec(b)−vec(a) 的结果向量最相似的词向量，其中 vec(w) 为 w 的词向量。

```
def get_analogy(token_a, token_b, token_c, embed):
    '''
    @params:
        token_a: 词a
        token_b: 词b
        token_c: 词c
        embed: 预训练词向量
    @outputs:
        res: 类比词d
    '''
    vecs = [embed.vectors[embed.stoi[t]]
            for t in [token_a, token_b, token_c]]
    x = vecs[1] - vecs[0] + vecs[2]
    topk, cos = knn(embed.vectors, x, 1)
    res = embed.itos[topk[0]]
    return res
```

```
get_analogy('man', 'woman', 'son', glove)
# 'daughter'

get_analogy('beijing', 'china', 'tokyo', glove)
# 'japan'

get_analogy('bad', 'worst', 'big', glove)
# 'biggest'

get_analogy('do', 'did', 'go', glove)
# 'went'
```

##04_总结

词向量表示是许多NLP任务的基础，word2vec利用的是局部的信息而GloVe的方法是通过考虑了语料库中所有词的统计信息。使得模型能够获取word sense更加优异。

参考文献

Pennington J , Socher R , Manning C . Glove: Global Vectors for Word Representation[C]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.	glove	• Senmantic vectro space models of language represent each word with a real-value vector
		• Most word vector methods rely on the distance or angle between pairs of vectors as primary method for evaluating the intrinsic quality of such a set of word representations.
		• Two main model families for learning word vector
		  • Global matrix factorization, such as latent semantic analysis(LSA)
		  • Local context window,such as skip-gram

Arora S , Li Y , Liang Y , et al. RAND-WALK: A Latent Variable Model Approach to Word Embeddings[J]. Computer Science, 2015:1242-1250.	A latent variable...	
Yin Z , Shen Y . On the Dimensionality of Word Embedding[J]. 2018.	7368-on-the...	
Arora S , Li Y , Liang Y , et al. Linear Algebraic Structure of Word Senses, with Applications to Polysemy[J]. 2016.	Linear Algebrai...	• Word embeddings are contructed using Firth's hypothesis that a word's sense is captured by the distribution of other words around it • Classical vector space models use simple linear algebra on the marix of word-word co-occurrence couts, whereas recent neural network and energy-based models such as word2vec use an obejctive that involves a nonconvex function of the word co-occurrences • Describing how multiple senses of a word actually reside in linear superposition within the standard word embeddings word2vec and GloVe。
Levy O , Goldberg Y , Dagan I . Improving Distributional Similarity with Lessons Learned from Word Embeddings[J]. Transactions of the Association for Computational Linguistics, 2015, 3:211-225.	Improving Distribut...	
	Evaluation methods...	