Text summarization part 1

文本摘要应用场景：QA，论文摘要，上市公司的报告，舆(情)论控制，Newsletter

algorithm engineer：算法基础，代码功底，业务理解和应用能力

良好的的产品sense和业务owner意识

Language

Denotational Semantics meaning

Wordnet：① Great but missing nuance ② Missing new meaning of word
③ subjective ④ requires human labor to create and adapt
⑤ Can't compute accurate word similarity

Representing words as discrete symbols：

ex: one-hot encode

problems：① 词表大且增长快 ② 很难表示相近的含义：motel, hotel
③ 不具有通用性

用上下文表达它们的含意

N-gram：N=1 unigram
N=2 bigram
N=3 trigram

N-gram LM

problem of N-gram: 数据稀疏，难免会出现OOV的问题 (OOV: out of vocabulary)
随着n的增大，参数空间呈指数增长(难度实难)
缺少长期依赖，只能建模到前n-1个词
无法表示一词多义(语义鸿沟)

word vector => also called word embeddings or (neural) word representations

Semantic and syntactic relationships ex: King & Queen 的距离是很接近的

Neural Probabilistic language Model (2003)

Problem of NPL model: 有限的前文信息　计算量过大　词向量是副产品
↓
在Softmax和tanh中间的全连接层计算量很大

Word2Vec:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} P(w_{t+j} | w_t; \theta)$$

objective function $J(\theta)$　$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \neq 0} \log P(w_{t+j} | w_t; \theta)$

minimize objective function <=> Maximizing predictive accuracy

how to calculate $P(w_{t+j} | w_t; \theta)$?
Answer: use two vectors per word w:
v_w when w is a center word

u_w when w is a context word

$$P(O|C) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

→ Dot product compares similarity of o and c. $u^T v = u \cdot v = \sum_{i=1}^{n} u_i v_i$

Larger dot product = larger probability

Normalize over entire vocabulary to give probability distribution

ex: softmax function $\mathbb{R}^n \to (0,1)^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

"max" because amplifies probability of largest x_i → return a distribution

"soft" because still assigns some probability to smaller x_i

frequent used in DL

Word2vec:

① CBOW 多个input周围字来预测一个中心词

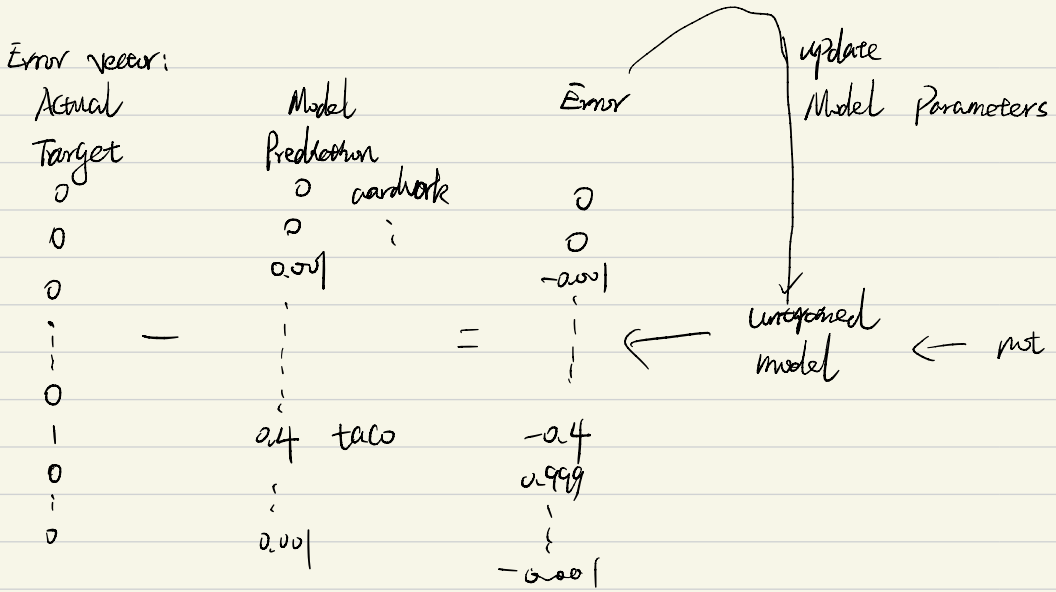
② skip-gram 一个中心词来预测周边的单词

Extract first sample

not ——→ Untrained Model

Task:

Predict neighboring word



Diff between skip-Gram & CBOW:

CBOW is simpler & faster; skip-gram学习到的语义更多
CBOW更倾向学习同义的语义

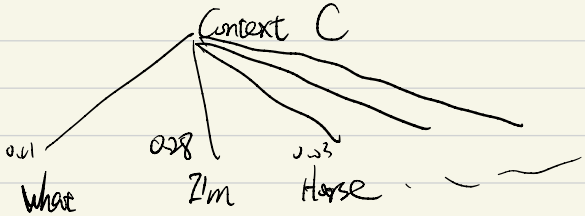
Problem of word2vec:

类别太多计算量大

⇓

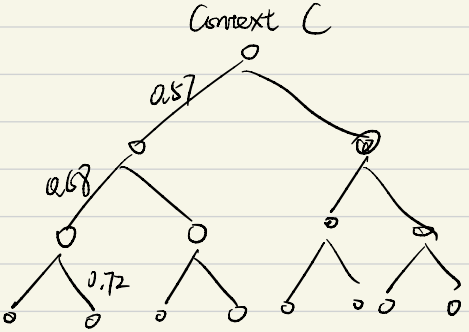
解决方案：Hierarchical Softmax (H-Softmax)

Switch Softmax to Binary Tree



You must compute all of the terminal leaves

O(n) to O(log(n))



P(I'm|C) = 0.57 · 0.68 * 0.72 = 0.28

P(w|C) what I'm Horse

Encode Node

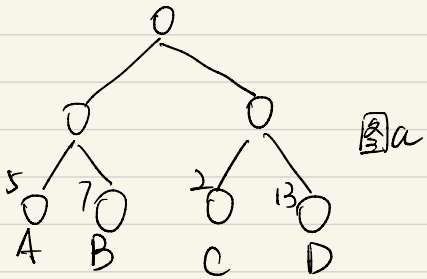
Construct Tree & Huffman Encoding

- Random construct

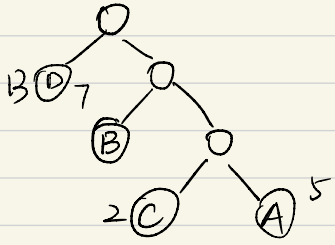
↓

Term Frequency Huffman Tree

Advantage for Huffman Encoding



图a



图b

带权路径长度

图a: WPL=5*2+7*2+2*2+13*2=54

图b: WPL=5*3+2*3+7*2+13*1=48

这类方法都是 approximating the Softmax，大约提高了50X的速度

Problem of Hierarchical Softmax:

当预测的词是生僻词时(频率使用比较低的时候)走的路径很长，计算量非常大

Negative sampling(负采样):

→在字符的最中，这embedding一般是已经训练好了的

Softmax: 1) look up embedding 2) calculate prediction

3) Project to output vocabulary

Switch to logistic regression Task:

一个想法是利用上下文来预测:

ex:

not → [Untrained Model Task: Predict neighboring word] → thou

To:

not →
thou →

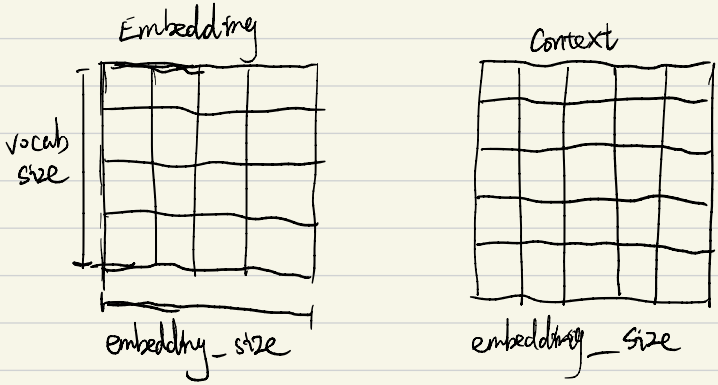
Untrained Model
Task:
Are the two words neighbors?

→ 0.90 (预测这对概率，最好接近于1)

怎么进行负采样：从vocabulary中而不是上下文的单词中拿出来单词，构建一个负样本。

skip-Gram with negative Sampling (SGNS):

start with two Random Matrices



input word	output word	target	input * output	sigmoid( )	Error( )
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68

target - sigmoid( )

超参数 Hyperparameters gensim default是5：window size

Smaller window sizes (2-15) lead to embeddings where high similarity scores between two embeddings indicates that the words are interchangeable

Large window sizes (15-50, or over more) lead to embeddings where similarity is more indicative of relatedness of the words.

实际情况是可以通过 word tree 来看 embedding 的关系

Negative samples: gensim 中 default 为 5

正常情况下，语料或数据集越大，negative sampling 就越小；反之亦然