2024年1月12日 23:51

期中考试一题一分，recess week。
期末大作业要有创新，写张纸来解释
同行评审算平均分有个门槛
第4，10周没有mini coding，所以剩下的minicoding分比较高
有请假情况在ntulearn上提交official letter

IRA/TRA 5questions,多选，10分钟，有密码

preprocessing

2024年2月2日 星期五 15:49

RE
BOW
Stem
Lemmatazation

Term weighting, topic modeling, dimension reduce

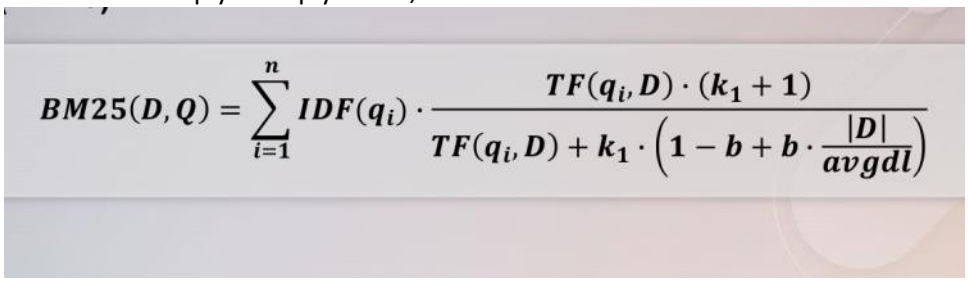
2024年1月12日 23:51

1. Term weighting
   Try to find what is most important and what is less important
   TF-IDF: TF-IDF = TF*IDF
   When TF-IDF is higher, the word is more important
   One way is Make it in ratio form: tf/words in doc
   Another way is Scale it : 1 + log(tf)

   On the other hand, idf = 1 + log[(1 + (N= docs in corpus))/(1 + (count = docs have term))]

   Each term TF-IDF is a dimension of VECTOR of one doc.
   Cosine similarity
   PROS: efficiency/ agnostic/ no supervised
   CONS: no semantics, so some frequent words with less meaning, size is challenge

2. BM25
   Consider probabilistic
   Rather than simply multiply them , BM25 use this



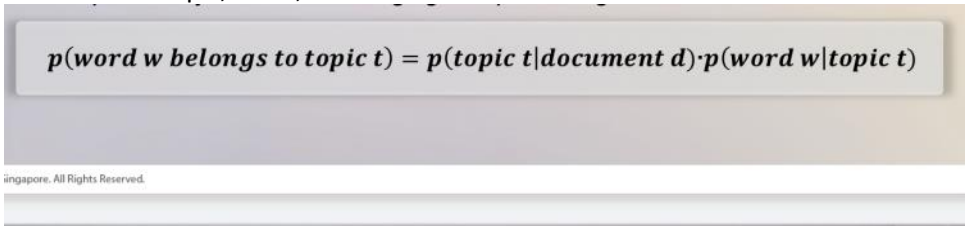
$$BM25(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i,D) \cdot (k_1+1)}{TF(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}$$

   |D| is length
   Avgdl is average doc length
   K1 for frequency saturation, and b for doc length normalization
   PROS: better than simple TF-IDF, robust since saturation is solved
   CONS: base on BOW, no semantic, not fit for sparse file and short file.

Topic Modeling

No supervised, for classification, find struct

1. LDA迪利克雷分配
   Define some topic, BOW, and then find out :



$$p(word\ w\ belongs\ to\ topic\ t) = p(topic\ t|document\ d) \cdot p(word\ w|topic\ t)$$

   Use gensim to calc LDA:
   Remove punctuation, convert lowercase, remove stopwords, use 7 topics to classify

   PROS: dim reduce, interprete
   CONS: no semantic, preprocessing sensitive

2. LSA潜在语义分析
   Try to find semantic structure and reduce dim.
   Doc-Term matrix, using TF-IDF, decomposed into sub.
   SVD
   Sub is word assign to topics, topic importance, topic distribution across docs.
   Also term-topic, topic-topic, topic-doc
   PROS: dim reduce, semantic
   CONS: no interpret, preprocessing

3. pLSA
   With probabilistic

$$P(w,d) = P(d)\sum_t P(w|t)P(t|d)$$

   LSA: scalable
   pLSA: less scalable, more interpretability

4. NMF非负矩阵分解
   Decomposed to words-topics, topics-docs.
   PROS: discover fundamental components, noise reduction
   CONS: fall into local minima, preprocessing

Summary: LSA : reduce dim, improve retrieval
LDA : probability, recommendation, clustering
NMF: suit for non-negative, sparse

Dimensionality Reduction

1. PCA: 主成分分析
   Initial variance = remaining var + lost var
   Covariance matirx eigenvector is u1.
   Steps: 1. standardise 2. covariance mat 3. eigen 4. doc product
   PROS: visual, other feature
   CONS: lack interpret, only linear

2. SVD:奇异值分解
   PROS: semantic, interpret
   CONS: need scale, may loss or overfitting when select dims.

Evaluation Metrics

2024年1月12日 23:51

Accuracy所有预测中预测正确比例
Precision预测的阳性中真阳性比例
Recall找到的阳性在所有真实阳性占比

Micro所有类别统一计算precision
Macro所有类别各自计算precision后平均

Neural network

2024年1月12日 23:51

Limitation of traditional models: no mem, no order, fix length

RNN: have loopback, new hidden state from prev hidden state. param all across entire network

Activation func: sigmoid, cons no zero, vanishing gradients.
tanh, cons vanishing gradients.

RNN used for classification, image caption, POS tagging, NER
cons of RNN: vanishing or exploding gradients, short mem.

LSTM, one of a RNN, have forger input output gate, solve vanishing gradient.

forgetgate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

inputgate

The input gate formula is given by:

$$i_t = \sigma(U^{(i)} x_t + W^{(i)} h_{t-1} + b^{(i)})$$

newcell

The equation for temporary new cell content is given by:

$$\tilde{c}_t = \tanh(U^{(c)} x_t + W^{(c)} h_{t-1} + b^{(c)})$$

The cell state is updated by a combination of the results from the forget gate and the input gate.

outputgate

- The output gate equation is given by:

$$o_t = \sigma(U^{(o)}x_t + W^{(o)}h_{t-1} + b^{(o)})$$

- The hidden state is given by:

$$h_t = o_t \tanh(c_t)$$

GRU is simplified LSTM
resetgate

The information from the previous hidden state and the current input passes through a sigmod:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

updategate

The formula for the update gate is given by:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

Bi-directional RNN:
LSTM and GRU are commonly used in Bi-RNNs.

Transformer

2024年1月12日 23:51

Encoder: typically rnn/ lstm/ gru
Aim to condense info to context vector.

Attention mechanism:
Queries Keys Values

q = St-1
q dot Key
then softmax(normalize, weight, sum to 1)
then product val and sum

aim to calc how relative to other words

encoders have a set of hidden state
decoders have each time hidden state
hidden state is used as query
q = current decoder hidden state
k = every encoder hidden state

sum concat to decoder input and for recurrent unit

Steps:

1. tokenization
2. input embeddings
3. positional encoding
4. multihead attention
5. add norm
6. feed forward
7. add norm
8. to encoder-decoder multihead
9. masked multihead attention
10. add norm
11. multihead attention
12. feed forward
13. fc linear
14. softmax
15. output

use sin func to ensure that pos is consistent across different sequence lengths

2 linear transformation 1 activate func, enhance representation independently

q comes from previous decoder layer

- Transformer Models:

– Utilise self-attention mechanisms to process entire input sequences simultaneously, providing
efficiency and effectiveness in capturing long-range dependencies.
– Feature an encoder-decoder architecture with multi-head attention, allowing the model to capture a
diverse range of information and relationships within the data.
– Do not rely on recurrence or convolution, making them well-suited for parallel processing and handling
sequences with complex structures.

大纲

2024年3月5日 21:21

1. NLP概述--用于预处理文本，剔除不必要的字符和无用词汇，简化词汇，BOW不考虑语义
2. 语言特征提取--NER，POS，DP层层递进考虑词在句中作用，为后续处理铺路
3. 词权重和主题建模
4. 传统NLP
5. 评估指标和词嵌入

1. 预处理，词袋， Ngram

2024年3月5日 21:21

PPT:

Preprocessing:
RegEX

Character	Description	Examples
[ ]	A set of characters	“[a-m]”
\	Special Sequence/Escape	“\.”
.	Any character (except newline)	“c.t”
^	Starts With	“^hello”
$	Ends With	“planet$”
*	Zero or more occurences	“he.*o”
+	One or more occurences	“he.+o”
?	Zero or one occurences	“colou?r”
{}	Exactly the specified no. of occurences	“he.{2}o”
|	Either or	“start|stop”
()	Matches exact characters	“(hello)+”

Character	Description	Examples
\A	Returns a match if the specified characters are at the beginning of the string	“\AThe”
\b	Returns a match where the specified characters are at the beginning or at the end of a word	“\bain” | “ain\b”
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	“\Bain” | “ain\B”
\d	Returns a match where the string contains digits (numbers from 0-9)	“\d”
\D	Returns a match where the string DOES NOT contain digits	“\D”
\s	Returns a match where the string contains a white space character	“\s”
\S	Returns a match where the string DOES NOT contain a white space character	“\S”
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	“\w”
\W	Returns a match where the string DOES NOT contain any word characters	“\W”
\Z	Returns a match if the specified characters are at the end of the string	“Spain\Z”

Character	Description
[ntu]	Returns a match where one of the specified characters (n, t, or u) is present
[a-d]	Returns a match for any lower-case character alphabetically between a and d
[^ntu]	Returns a match for any character EXCEPT n, t and u
[0123]	Returns a match where the specified digits (0, 1, 2 or 3) are present
[0-9]	Returns a match for digits between 0-9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case
[+]	Returns a match for the char '+" (no special meaning here)

Tokenization, Stemming, Lemmatization(return dictionary form, consider the context), BOW(easy to measure distance between 2 vectors), N-grams(rely on N precedes words)

Code:

```
words=word_tokenize(string)
stemmed_string = reduce(lambda x, y: x +" "+ps.stem(y), words, "")
```

```
lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))
```

```
import nltk
import re
import numpy as np
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
text = '''
Never gonna run around and desert you.
'''
dataset= nltk.word_tokenize(text)
for i in range(len(dataset)):
    dataset[i] = dataset[i].lower()
    dataset[i] = re.sub(r'\W', ' ', dataset[i])
    dataset[i] = re.sub(r'\s+', ' ', dataset[i])#remove all the white spaces
filtered_sentence = [w for w in dataset if not w.lower() in stop_words]
print(filtered_sentence)
```

Quiz:

BOW has a FIXED vector length

BOW converts unstructured text into structured data

A word's part-of-speech tag is taken into consideration during lemmatization but not for stemming

Morphological analysis is conducted on each word for lemmatization

InClass:

2. 语言分析和信息提取

2024年3月6日 21:39

PPT:

NER命名实体，是从无结构的text中来，而不是从有结构的data中来，可能多个词组成一个命名实体

作用: sentiment analysis, deal large text, answer detection, info retrieve

POS tagging:Tagging can be done using linguistic patterns, context and predefined dictionaries(static rules), and using probability models, such as Hidden Markov Models(stochastic model).

HMM隐马尔可夫模型，拥有states和observations，概率有transmission和emission

emission 是指某个词性可能会发射出的单词概率，是根据训练集的先验概率来的，

transmission指的是从<s>标签开始的各个词性一直连接到<e>标签的概率

根据所有概率的乘积确定最可能的输出，不适用于过长的句子。

所以开始使用Viteribi算法进行优化，根据构建的树，反向遍历最优路线。

作用： 分类，翻译，生成

依存句法分析： 考虑更多，词汇在句中的成分，主谓宾等。

Code:

SpaCy用法

Pandas

NER POS:

```
nlp_en_model=spacy.load("en_core_web_sm") #construct a pipeline
doc1=nlp_en_model(raw_text)#this instance is callable and returns a Doc object
for span in doc1.ents:#doc.ents have many spans, each span is a named entity, and each
span has a label_.
    print(span.text, span.label_)
for token in doc1:
```

```
    print(token.text, token.pos_, token.dep_)#token is a word, pos_ is part of speech,
dep_ is dependency
```

DP:

```
import spacy
from nltk import Tree
def tok_format(tok):
    return "_".join([tok.orth_, tok.dep_])
def to_nltk_tree(node):
    if node.n_lefts + node.n_rights > 0:
        return Tree(tok_format(node), [to_nltk_tree(child) for child in node.children])
    else:
        return tok_format(node)
nlp = spacy.load('en_core_web_sm')
doc = nlp(u'She took the lesson to heart.')
[to_nltk_tree(sent.root).pretty_print() for sent in doc.sents]
```

Quiz:

区分 word_tokenize和sent_tokenize
Doc.sents[n].ents[m].label_
注意[]的构造对象

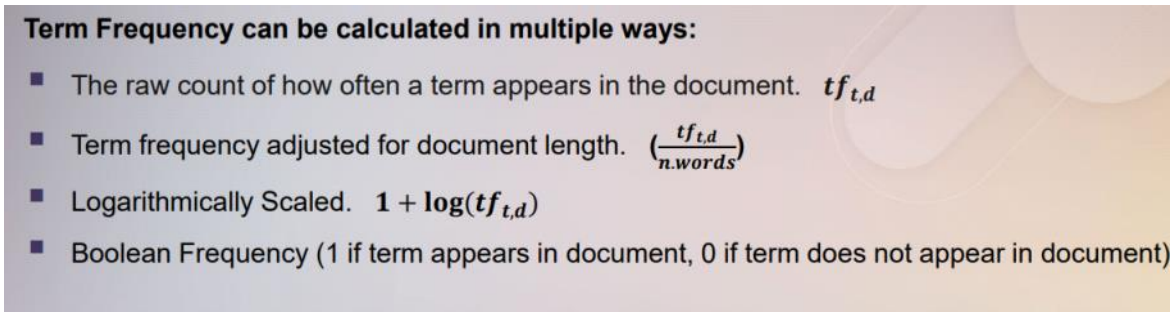
3. 词权重 主题建模 降维

2024年3月7日 3:57

PPT:

1.词权重

TF-IDF:
if not relevant, TF-IDF score is 0.
TF



IDF

$$idf(t, D) = log\left(\frac{1+N}{1+count(d \in D: t \in d)}\right) + 1$$

it is the ratio of docs/docs, if a word is popular in many docs, gets a lower idf score.
DOC have a vector that represents each word TF-IDF score.
DOC to DOC similarity can be calculated from cosine similarity of vectors.
优势：效率，通用语言，无监督
缺点：无语义，偏见，基于词汇量

BM25: best match
consider doc length normalization and prob distribution, k and b is for tuning.
优点：适用于大文本的信息提取，更加平衡的词频
缺点：还是无语义，稀疏数据表现不佳

2.主题建模

unsupervised/ co-occurrence pattern with unstructured text

LDA: 隐含迪利克雷分布
base on BOW, need pre topics,
是根据先验和采样确定后验，分布服从迪利克雷分布和多项式分布

优点：降维，直观，适合聚类任务
缺点：无语义，需要预处理

LSA:潜在语义分析
use TF-IDF for doc*term matrix, then use SVD reduce dim.
SVD: convert to doc-topic *topic-topic *topic-term
优点：降维，语义
缺点：不直观，需要预处理

pLSA:基于概率
认为每篇文章由多种主题不同概率混合，同理词汇概率也是不同主题下概率求和
扩展性差，但相对更直观。

NMF:非负矩阵分解
use TF-IDF
factorize: word-topic* topic-doc
用两个评估函数来评估估计值的好坏
用于img 和text mining，提取数据主题
优点：提取基础成分，降噪
缺点：非凸优化，需要预处理

3.降维

PCA: 重构坐标系，聚焦于主成分，主要步骤 标准化（0均值，1方差）协方差矩阵，特征向量，用少数主要特征向量重构
优点：降维到3D就可视化，发现线性组合的特征
缺点：不直观，线性假设。

SVD： convert to U, Σ，V^T
kSVD: consider k largest singular values
优点：有语义，直观
缺点：规模敏感，保留维度敏感

Quiz:

PCA和SVD对比，是PCA构建超平面
BM25消除了文档长度影响
IDF是文集中越常见的词分数越低
只有SVD有语义，并且直观，PCA是基于线性假定的。

LDA只使用概率模型，LSA使用了SVD也就是分解3矩阵，NMF分解2矩阵

Code:

```
TfidfVectorizer.fit_transform(df1.iloc[0])
```
此举返回的是term-doc的稀疏矩阵，与之相应的fit返回model本身，

BM25的输入需要token化的

```
bm25 = BM25Okapi(tokenized_corpus)

doc_scores = bm25.get_scores(tokenized_query)
```

使用LDA时候
```
gensim.models.LdaMulticore()
```
需要指定文集corpus，来源于
```
gensim.corpora.Dictionary(data_words).doc2bow()
```
以及其id2word，就是corpora.Dict

```
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
```

bow或者说文集实际上是一个term doc词频的矩阵

LSA模型有着极其类似的用法

```
lsa_model = LsiModel(doc_term_matrix, num_topics=num_topics, id2word = id2word)
```

做NMF之前还是先计算TF-IDF,正如ppt中所说。

```
# Create an NMF instance: model
# the 10 components will be the topics
model = NMF(n_components=10, random_state=5)

# Fit the model to TF-IDF
model.fit(X)

# Transform the TF-IDF: nmf_features
nmf_features = model.transform(X)
```

4. 传统NLP

2024年3月7日 6:15

PPT:

Text Classification:

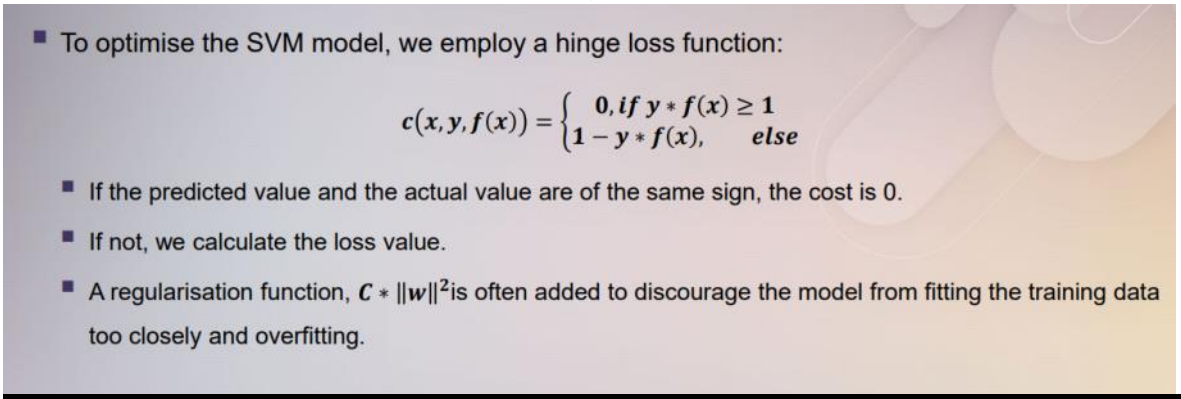
监督学习

硬规则分类，概率分类

朴素贝叶斯，支持向量机，极限学习机，高斯过程，线性回归。

朴素贝叶斯是词袋技术，通过先验和似然求后验，使用拉普拉斯平滑，加1或者加Vocab大小

支持向量机：用于二分类的线性非线性分类问题，最大化hyperplane和数据集的边距，使用TFIDF作为向量



kernel trick 用于处理本来难以线性分割的数据，使用点积代替直接计算高维空间中的点，RBF，SIGMOID等，核函数用linear是二分类问题

极限学习机只有1层隐前馈网络，没有反向传播环节。参数随机初始化，隐藏层的参数不直接更新，而是通过输出层权重的计算间接影响。

高斯过程，先计算均值m和协方差矩阵k，也叫kernel mat，gram mat，必须为正值和半定，类似SVM一样有核函数，比如RBF来映射数据，超参L^2控制函数陡峭程度

线性回归，拟合一个直线，学习率a是梯度下降过程中的激进程度

Clustering:

无监督学习，一般基于距离或者相似性度量

K-means

随机初始化，然后迭代

分层聚类

形成分层树，可以自顶向下或者自底向上，有一个ward linkage计算差异度有效避免噪声

模糊聚类

每个数据点都有归属某个聚类的概率标注

Code:

```
#Create a svm Classifier
clf = svm.SVC(kernel='rbf') # RBF kernel
#Train the model using the training sets
clf.fit(train_review_tfidf, train_sent)
#Predict the response for test dataset
y_pred = clf.predict(test_review_tfidf)
```

5. 评估指标和词嵌入

2024年3月7日 6:15

PPT:

评估指标：

以下是对于分类器的指标：：：

confusion matrix:
accuracy(how many correct in predictions)
precision(how many correct in positive predictions)
recall(how many really positive are found)

微观指标，所有分类结果混在一起总共计算一个最终指标
宏观指标，各自计算完指标，再平均，各个类视为相同重要性

每个实例同等重要时候用微观，每个类同等重要时候用宏观

F1 Score
precision和recall的调和平均，为了处理极端情形，处理阳性数据的指标

AUCROC
AUC=0.5时候代表随机分类器

以下是对于生成器的指标：：：

BLEU counts the number of n-grams that appear in both the generated sentence and the target sentence. Common n-gram sizes used are 1, 2, 3, or 4.使用裁剪精度避免重复单词假象，对1234gram的精度采用几何平均，然后用brevity惩罚太短的句子。
专注于precision

ROUGE用于文本概括的评估，
rouge-N评估预测和参考的重复Ngram，
rouge-L评估最长字串
rouge-s skip-bigram
专注于recall

METEROR考虑单词序列评估翻译质量，计算F1Score和chunk penalty的乘积

词嵌入：

比如bow和tfidf，为了解决语义问题和维度问题，引入：
word2vec和GloVE

word2vec基于一个思想，相似语境有相似含义。使用CBOW和Skip-gram
CBOW预测目标词汇，Skipgram反过来预测context，skipgram可以达到20context
CBOW效率更高，skipgram处理大量语境
使用滑动窗计算概率，更新神经网络的迭代
训练过程中，输入输出相似度用点积表示，标量积，使用sigmoid
只需要最终的词嵌入输出，上下文嵌入被丢弃

softmax只是一个归一化的概率表征

negative sampling优化计算效率，只抽取2到20个词，并且数据集越大抽取越少

GloVE

无监督的

计算共现矩阵，计算共现频率

Code:

对一个逻辑回归训练模型进行ROCAUC评估

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import matplotlib.pyplot as plt
#instantiate the model
log_regression = LogisticRegression()
#fit the model using the training data
log_regression.fit(train_review_tfidf, train_sent)

y_pred_proba = log_regression.predict_proba(test_review_tfidf)[::,1]
fpr, tpr, _ = metrics.roc_curve(test_sent,  y_pred_proba)
auc = metrics.roc_auc_score(test_sent, y_pred_proba)
```

计算BLEU分数

```
import nltk
from nltk import word_tokenize
from nltk.translate.bleu_score import SmoothingFunction
ref = 'The guard arrived late because it was raining.'
cand = 'The guard arrived late because of the rain.'
smoothie = SmoothingFunction().method1
reference = word_tokenize(ref)
candidate = word_tokenize(cand)
weights = (0.25, 0.25, 0.25, 0.25)
BLEUscore = nltk.translate.bleu_score.sentence_bleu([reference], candidate, weights,
smoothing_function=smoothie)
print(BLEUscore)
```

计算ROUGE

```
rouge.compute
```

类似的有METEOR

```
score=round(meteor([word_tokenize('The cat sat on the mat')],
                   word_tokenize('The cat was sat on the mat')), 4)
```

Word2Vec

```
# Tokenize the sentences into words
tokenized_sentences = [word_tokenize(sentence.lower()) for sentence in sentences]
# Train a Word2Vec model
model = Word2Vec(tokenized_sentences, vector_size=100, window=5, min_count=1, sg=0)
```

Quiz:

二分类问题中激活函数用sigmoid，reLu可能常用于深度学习，softmax一般是多分类问题

二分类问题最后全连接层就1个输出节点，多分类多节点

Hyperparameter Tuning

2024年1月12日 23:51

external config

K-Fold交叉验证：分为K组，只有其中一组是测试集，训练K次，取MSE
K取值较为重要，每组数据集需要有统计特征。固定K=10，或K=n，需要每个采样都能当一次测试集

优化器：用于最小化损失函数，迭代地更新W和b，有SGD，Adam，RMSprop

梯度下降：寻找局部最小，运算量大
随机梯度下降：每次迭代选择打乱后的随机子集来训练，引入更多噪音，需要更多的迭代，运算量略小。

Adam自适应矩估计：是SGD的拓展，动态学习率

RMSProp均方根传播：相较于G D微调更少，但需要指定学习率。

损失函数：根据实际问题选择合适的函数，分类问题的损失函数：binary cross entropy和categorical cross entropy，
CCE一般和softmax一起使用

batch size，越少的batchsize更新越频繁，大batchsize用于性能较强电脑

学习率：一般从0.1到0.0001

Epoch：轮数，提前终止回调

梯度剪切：比如RNN可能发生梯度爆炸
Regularization：解决overfitting，随机dropout只在训练过程起作用，推理过程不drop

Transformer

2024年1月12日 23:51

Pre-train: large/ unsupervised
Fine-tuning: further on low level/ downstream tasks/ supervised

aim to create universal pre-training model
GOOGLE T5: treat all NLP as generation
GPT3 : pre-train good.

objective AR and AE, AR for generating AE for understanding

AR	AE
generate	understand
need decoder	bidirection
limited L2R	mask token
XLNet	discrepancy to different use

transformer XL use relative pos coding

2 sub word tokenizers: WordPiece/ BPE

Bert pretrain: denoising
Bert finetuning: SQuAD/ classification/ NER

RoBERTa more advanced: use BPE, change mask
DistilBERT

GPT1 only decoder stacks.

use InstructGPT supervised learning and RLHF

T5 use BERT masking

Facebook BART: use T5 denoising, 30% token masked

XLNet

▪ Encoder or decoder-based architectures and pre-training objectives. ▪ Autoencoder (AE) models excel in bi-directional context understanding. ▪ Autoregressive (AR) models have an advantage in that there is less discrepancy between pre-training and fine-tuning downstream tasks.