用户诉求热词/主题挖掘

项目背景

基于用户提交工单的文本信息，挖掘用户主要诉求信息。任务分为两个：热词提取（用于销售部门进行展示）、主题挖掘（用于后端运营进行客户分析，制定后续市场策略）

项目说明

该文档记录的部分，仅包含业务应用层面的实现，模型训练及调优后续在模型文档中

案例：用户诉求raw text一条

> 客户之前已反馈问题XXX地点无法上网，但是手机上显示的是有信号接入的，工单号：xxx，客户未接到电话，通过工单处理结果告知用户，用户此问题还为解决，要求再次处理该问题，如客户未接到电话请多次拨打

使用技术

1. 热词挖掘任务 + 展示

- jieba：用于中文文本切分，关键词，停词字典管理
- sklearn：使用feature_extraction.text中的TfidfVectorizer进行关键词挖掘（tfidf算法）
- wordcloud：进行热词挖掘的展示

2. 诉求主题挖掘

- gensim: LDA模型
- jieba：同上

热词挖掘任务

业务内容文本切割

```
contents_list = [str(x).upper() for x in df['业务内容']]
```

关键词字典添加，文本分割

```
# %%
'''
在jieba中加入专用词库
'''
add_word_ls = ['xxx']
for word in add_word_ls:
    jieba.add_word(word)
# 切分
wd = []
for d in contents_list:
    word_list = jieba.lcut(d)
    wd.append(' '.join(word_list))
```

TF-IDF算法，热词挖掘

```
'''
TF-IDF algorithm
'''
from sklearn.feature_extraction.text import TfidfVectorizer
# 初始化
tfidf = TfidfVectorizer()
# 向量化
weight = tfidf.fit_transform(wd).toarray()
word = tfidf.get_feature_names()

# 定义热词词典
word_dict = {}

# 热词权重
for i in range(len(weight)):
    for j in range(len(word)):
        if word[j] in word_dict:
            word_dict[word[j]] += weight[i][j]
        else:
            word_dict[word[j]] = weight[i][j]
```

热词文字云展示

词频转换，词云转换

```
# make a frequency dict
word_freqence = {x[0]:x[1]*100 for x in df_tfidf.head(100).values}
```

文本词云展示

```
# change figure size
matplotlib.rcParams['figure.figsize'] = (10.0, 5.0)

# generate the word cloud
# default font path: '"c:/windows/Fonts/simhei.ttf"'
wordcloud = WordCloud(font_path='c:/windows/Fonts/simhei.ttf',
background_color='white',
                      max_font_size=80)
wordcloud = wordcloud.fit_words(word_freqence)
plt.axis('off')
plt.imshow(wordcloud)
```

诉求主题挖掘

诉求文本切分

```
raw_documents = []
for text in df["业务内容"]:
    raw_documents.append(text[text.find("用户诉求")+5:])
```

LDA模型

```
# 词性flag
flags = ('n', 'nr', 'ns', 'nt')

# 文本列表
words_ls = []
for text in raw_documents:
    words = [w.word for w in jp.cut(text) if w.flag in flags and w.word not in
stopwords]
    words_ls.append(words)

# 字典
dictionary = corpora.Dictionary(words_ls)

# 向量化
corpus = [dictionary.doc2bow(words) for words in words_ls]

# lda模型
lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary, num_topics=12)

for topic in lda.print_topics(num_words=10):
    print(topic)

# 打印主题结果
print(lda.inference(corpus))
```