基于TF-IDF的新闻推荐算法的简单实现

组员1：肖文璇 学号：2018202163

组员2：雷雨寒 学号：2018202173

基本原理与思路

TF-IDF算法

TF-IDF（term frequency–inverse document frequency）是一种用于资讯检索与资讯探勘的常用加权技术。TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜寻引擎应用，作为文件与用户查询之间相关程度的度量或评级。除了TF-IDF以外，因特网上的搜寻引擎还会使用基于连结分析的评级方法，以确定文件在搜寻结果中出现的顺序。

在一份给定的文件里，**词频（term frequency，TF）**指的是某一个给定的词语在该文件中出现的频率。这个数字是对**词数**(term count)的归一化，以防止它偏向长的文件。对于在某一特定文件里的词语t_i来说，它的重要性可表示为：

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子中$n_{i,j}$是该词在文件中d_j的出现次数，而分母则是在文件d_j中所有字词的出现次数之和。

逆向文件频率 (inverse document frequency, IDF) 是一个词语普遍重要性的度量。某一特定词语的IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到：

$$idf_i = log\frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中

- $|D|$：语料库中的文件总数
- $|\{j : t_i \in d_j\}|$：包含词语t_i的文件数目（即$n_{i,j} \neq 0$的文件数目）如果该词语不在语料库中，就会导致被除数为零，**因此一般情况下使用**$1 + |\{j : t_i \in d_j\}|$

TFIDF的主要思想是：TFIDF实际上是：$TF * IDF$，如果某个词或短语在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

$$tfidf_{i,j} = tf_{i,j} * idf_i$$

基于内容相似度的推荐算法

基于内容，即把与**用户喜欢看**的新闻**内容相似**新闻推荐给用户。基于内容的推荐算法的主要优势在于无冷启动问题，只要用户产生了初始的历史数据，就可以开始进行推荐的计算。而且随着用户的浏览记录数据的增加，这种推荐一般也会越来越准确。

利用TF-IDF判断新闻相似性

找到两个新闻的共同关键词，分别计算该关键词在两篇文章的TF-IDF值（即该关键词在两篇文章的重要程度），然后将所得的两个TF-IDF值相乘，该乘积越高，则说明这个关键词在两篇文章中都很重要。最后求得所有共同关键词TF-IDF乘积的和，即得两篇文章的相似性。

公式如下：

$$Similarity(A,B)=\sum_{i\in m}TFIDF_{A_i}*TFIDF_{B_i}$$

其中m是两篇文章重合关键词的集合。

根据用户浏览历史构建用户喜好库

根据用户的浏览历史，我们可以得到一系列用户观看过的新闻，根据新闻的标签、标题、内容，我们可以提取出每个新闻的关键词，这些关键词构成了该用户的新闻喜好库，如果用户浏览历史中的关键词重复出现的次数越多，则该关键词的权重越高。

在对用户进行新闻推荐的时候，我们在总新闻库中计算每个新闻与用户的新闻喜好库的共同关键词的匹配程度，即

$$Similarity(news,user)=\sum_{i\in m}W_i*TFIDF_{user_i}*TFIDF_{news_i}$$

其中m是该新闻与用户喜好库中重合关键词的集合，W_i代表关键词i在用户喜好库中的权重。

数据获取与数据清理

MIND数据集

新闻推荐的MIND数据集是从Microsoft新闻网站中收集的数据，我们从中使用了用户的行为数据和新闻数据。

用户的行为数据 behaviors.tsv

behaviors.tsv文件包含浏览日志和用户的新闻点击记录：

- Impression ID：展示的ID。
- User ID：用户的匿名ID。
- Time：展示时间，格式为“ MM / DD / YYYY HH：MM：SS AM / PM”。
- History：在此浏览之前，该用户的新闻点击历史（点击新闻的ID列表）。点击的新闻文章按时间排序。
- Impressions：此浏览中显示的新闻列表以及用户对其的点击行为（1为点击，0为非点击）。浏览中新闻的顺序已被打乱。

示例：

Column	Content
Impression ID	91
User ID	U397059
Time	11/15/2019 10:22:32 AM
History	N106403 N71977 N97080 N102132 N97212 N121652
Impressions	N129416-0 N26703-1 N120089-1 N53018-0 N89764-0 N91737-0 N29160-0

新闻数据 news.tsv

包含详细的新闻信息：

- News ID - 新闻编号
- Category - 类别
- SubCategory - 子类别
- Title - 标题
- Abstract - 摘要
- URL - 网址

示例：

Column	Content
News ID	N37378
Category	sports
SubCategory	golf
Title	PGA Tour winners
Abstract	A gallery of recent winners on the PGA Tour.
URL	https://www.msn.com/en-us/sports/golf/pga-tour-winners/ss-AAjnQjj?ocid=chopendata

加载数据集

因为Mind数据集过大，我们在测试使用时只使用部分的Mind数据集。

```
#load news
news=pd.read_csv('../dataset/MINDsmall_train/news.tsv',sep='\t',
                         names=['News ID','Category','SubCategory','Title','Abstract','URL','Title Entities','Abstract Entities'])
news=news.dropna(axis=0,how='any')

#load behavior
behaviors=pd.read_csv('../dataset/MINDsmall_train/behaviors.tsv',sep='\t',
                                names=['Impression ID','User ID','Time','History','Impressions'])
behaviors=behaviors.dropna(axis=0,how='any')
```

下载新闻内容

光有新闻标签与摘要不足以获得完整的新闻关键词，因此利用BeautifulSoup，根据网址获得新闻内容。

```
for i in range(len(sr)):
    address='./news/'+str(i)+'.txt'
    f=open(address,"w",encoding='utf-8')
    r=requests.get(sr[i])
    soup=bs4.BeautifulSoup(r.content,"lxml")
    txt=str(soup.find(id='maincontent').text)
    txt=txt.replace('\n','')
    f.write(txt)
    f.close()
    print(i)
```

数据清理

在数据中发现有重复的行与空值，因此需要去除

1. 去除重复行

```
def remove_duplicates(df, cols):
    """ Remove duplicated entries.

    Args:
        df (pd.DataFrame): Pandas dataframe.
        cols (list of str): Name of columns in which to look for duplicates.

    Returns:
        df (pd.DataFrame): Pandas dataframe with duplicate rows dropped.

    """
    for col in cols:
        # Reset index
        df = df.reset_index(drop=True)

        # Find where the identifier variable is duplicated
        dup_rows = np.where(df.duplicated([col]) == True)[0]

        # Drop duplicated rows
        df = df.drop(dup_rows)

    return df
```

2. 去除含有空值的行

```
def remove_nan(df, cols):
    """ Remove rows with NaN values in specified column.

    Args:
        df (pd.DataFrame): Pandas dataframe.
        cols (list of str): Name of columns in which to look for NaN.

    Returns:
        df (pd.DataFrame): Pandas dataframe with invalid rows dropped.

    """
    for col in cols:
        # Convert any empty string cells to nan
        df[col].replace("", np.nan, inplace=True)

        # Remove NaN rows
        df = df[df[col].notna()]

    return df
```

3. 清除函数

调用`remove_duplicates`和`remove_nan`函数进行数据清理。

```
def clean_dataframe(df):
    """ Clean up the dataframe.

    Args:
        df (pd.DataFrame): Pandas dataframe.

    Returns:
        df (pd.DataFrame): Cleaned pandas dataframe.
    """

    # Remove duplicated rows
    cols = ["News ID"]
    df = remove_duplicates(df, cols)

    # Remove rows without values in specified columns
    cols = ['Category','SubCategory','Title','Abstract','URL','Title Entities','Abstract Entities']
    df = remove_nan(df, cols)

    return df
```

TfidfRecommender推荐器

TfidfRecommender推荐器是微软实现的一个基于sklearn中TfidfVectorizer构建的推荐器，它用于文章推荐，即根据一篇文章，搜索与它最相似的文章。有以下几个重要函数：

`tokenize_text`、`fit`、`__create_full_recommendation_dictionary`、`__organize_results_as_tabular`、`recommend_top_k_items`

我们在后文中会更改、利用到其中的一些函数实现基于内容的新闻推荐。

tokenize_text

输入：清理过后的文本

输出：分词后的文本 vectors_tokenized，文本向量化工具 tf (TfidfVectorizer)

1. 利用bert进行英文分词
2. 初始化文本向量化的工具，设置屏蔽一些停用词（TfidfVectorizer自带），如

```
['a', 'about', 'above', 'across', 'after', 'afterwards', 'again', 'against', 'all', 'almost']
```

```
    def tokenize_text(
        self, df_clean, text_col="cleaned_text", ngram_range=(1, 3), min_df=0
    ):
        vectors = df_clean[text_col]

        # Set vectorizer
        tf = TfidfVectorizer(
            analyzer="word",
            ngram_range=ngram_range,
            min_df=min_df,
            stop_words="english",
        )

        # Get appropriate transformer name
        bert_method = "allenai/scibert_scivocab_cased"

        # Load pre-trained model tokenizer (vocabulary)
        tokenizer = BertTokenizer.from_pretrained(bert_method)

        # Loop through each item
        vectors_tokenized = vectors.copy()
        for i in range(0, len(vectors)):
            vectors_tokenized[i] = " ".join(tokenizer.tokenize(vectors[i]))

        # Save to class variable
        self.tf = tf

        return tf, vectors_tokenized
```

fit

输入：分词后的文本

输出：文本的tf-idf特征矩阵

利用`tokenize_text`初始化的文本向量化工具，将文本转换成tf-idf矩阵，便于后续的分类检索。

```
def fit(self, tf, vectors_tokenized):
    """ Fit TF-IDF vectorizer to the cleaned and tokenized text.
    """
    self.tfidf_matrix = tf.fit_transform(vectors_tokenized)
```

__create_full_recommendation_dictionary

输入：清理过的文本

输出：包含与该文本的相似的所有文本的字典

根据前面得到的文本的tf-idf矩阵，计算文本库中所有文本与该文本的相似性，存入result中。

```
def __create_full_recommendation_dictionary(self, df_clean):
    # Similarity measure
    cosine_sim = linear_kernel(self.tfidf_matrix, self.tfidf_matrix)

    results = {}
    for idx, row in df_clean.iterrows():
        similar_indices = cosine_sim[idx].argsort()[: -(len(df_clean) + 1) : -1]
        similar_items = [
            (cosine_sim[idx][i], df_clean[self.id_col][i]) for i in similar_indices
        ]
        results[row[self.id_col]] = similar_items[1:]

    # Save to class
    self.recommendations = results
```

__organize_results_as_tabular

输入：包含与文本相似的所有文本的字典，k

输出：前k个与文本最为相似的文本，相似度得分等信息。

```
def __organize_results_as_tabular(self, df_clean, k):
    # Initialize new dataframe to hold recommendation output
    item_id = list()
    rec_rank = list()
    rec_score = list()
    rec_item_id = list()

    # For each item
    for idx in range(0, len(self.recommendations)):
        # Information about the item we are basing recommendations off of
        rec_based_on = list(self.recommendations.keys())[idx]
        tmp_item_id = str(
            df_clean.loc[df_clean[self.id_col] == rec_based_on][self.id_col].values[0])

        # Get all scores and IDs for items recommended for this current item
        rec_array = self.recommendations[rec_based_on]
        tmp_rec_score = list(map(lambda x: x[0], rec_array))
        tmp_rec_id = list(map(lambda x: x[1], rec_array))

        # Append multiple values at a time to list
        item_id.extend([tmp_item_id] * k)
        rec_rank.extend(list(range(1, k + 1)))
        rec_score.extend(tmp_rec_score[:k])
        rec_item_id.extend(tmp_rec_id[:k])

    # Save the output
    output_dict = {
        self.id_col: item_id,
        "rec_rank": rec_rank,
        "rec_score": rec_score,
        "rec_" + self.id_col: rec_item_id,
    }
```

recommend_top_k_items

调用`__create_full_recommendation_dictionary`和`__organize_results_as_tabular`得到前k个文本推荐。

```
def recommend_top_k_items(self, df_clean, k=5):

    if k > len(df_clean) - 1:
        raise ValueError(
            "Cannot get more recommendations than there are items. Set k lower."
        )
    self.__create_full_recommendation_dictionary(df_clean)
    self.__organize_results_as_tabular(df_clean, k)

    return self.top_k_recommendations
```

应用实例

根据如下一篇有关伊丽莎白的文章

	News ID	Category	SubCategory	Title	Abstract	URL	Title Entities	Abstrat Entities	cleaned_text
0	N55528	lifestyle	lifestyleroyals	The Brands Queen Elizabeth, Prince Charles, an...	Shop the notebooks, jackets, and more that the...	https://assets.msn.com/labs/mind/AAGH0ET.html	[{"Label": "Prince Philip, Duke of Edinburgh",...	[]	The Brands Queen Elizabeth Prince Charles and ...

我们可以得到以下最相似的5篇文章，都是关于“皇室”的文章

	rank	similarity_score	Title	Abstract	Category	SubCategory	URL
0	1	0.220413	Prince George's Royal Life in Photos	Photos of the future king of England who is third in line for the throne, behind his father and Prince Charles.	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AABDkOp.html
1	2	0.199708	Queen Elizabeth's Cousin Says Royal Family 'Don't Communicate Very Well'	Queen Elizabeth's Cousin: Royal Family 'Don't Communicate Very Well'	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAGCsB5.html
2	3	0.161782	It's Not All About the Corgis - Here Are the Royal Family's Other Beloved Pets	We all know how much Queen Elizabeth loved her Corgis.	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AADmbCD.html
3	4	0.158303	Cutest photos of the royal Cambridge kids	See the best photos of Prince George, Princess Charlotte and Prince Louis!	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAAM6VA.html
4	5	0.157612	What Do Prince George & Princess Charlotte Know About Their Royal Roles?	Do Prince William and Kate Middleton's kids know about their royal roles?	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAGeZKp.html

可以看到文章推荐的效果还是十分不错的。

根据用户偏好进行新闻推荐

初始化推荐器

利用前面所说的TfidfRecommender初始化推荐器，将id列设为New ID，选择用scibert方式分词

```
#load recommender
recommender = TfidfRecommender(id_col='News ID', tokenization_method='scibert')
```

根据用户偏好进行推荐

1. 对新闻数据进行清理
2. 利用`fit`函数，将所有新闻用TF-IDF矩阵的形式表示
3. 对新闻库中的每篇新闻，利用`recommender_top_k_items`在新闻库中找到与每篇最相似的前5篇新闻，以及它们的相似度分数。
4. 对于用户阅读的每篇历史新闻，提取它们在新闻库中最相似的前5篇新闻和分数，重复出现的新闻相似度分数会叠加。
5. 将这些得到的相似新闻根据重新计算的相似度分数排序，得到一个基于用户偏好的新闻推荐。

```
def user_recommendation(recommender,news,reader):
    #data preparation
    news=clean_dataframe(news)
    news=news.reset_index()[0:1000]

    #fit in tfidf recommender
    tf,vectors_tokenized=recommender.tokenize_text(news,text_col='Abstract')
    recommender.fit(tf,vectors_tokenized)
    topk=recommender.recommend_top_k_items(news,k=5)
    cols_to_keep=['News ID','Title','Abstract']

    #generate result
    rec=pd.DataFrame()
    like1=reader1.like
    for i in range(len(like1)):
        rec=rec.append(recommender.get_top_k_recommendations(news,like1[i],cols_to_keep))
    rec=rec.reset_index()

    drop=[]
    for i in range(len(rec)):
        for j in range(i+1,len(rec)):
            if rec['News ID'][i]==rec['News ID'][j]:
                rec['similarity_score'][i]=rec['similarity_score'][j]+rec['similarity_score'][i]
                drop.append(j)
    rec=rec.drop(drop)

    rec=rec.sort_values(by=['similarity_score'],ascending=False)
    rec['rank']=range(len(rec))
    rec=rec.reset_index()
    rec=rec.drop(columns='index')
    rec=rec.drop(columns='level_0')
```

测试新闻推荐结果

模拟一个读者，他的浏览历史都是与皇室相关的新闻：

	rank	similarity_score	Title	Abstract	Category	SubCategory	URL
0	1	0.220413	Prince George's Royal Life in Photos	Photos of the future king of England who is third in line for the throne, behind his father and Prince Charles.	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AABDkOp.html
1	2	0.199708	Queen Elizabeth's Cousin Says Royal Family 'Don't Communicate Very Well'	Queen Elizabeth's Cousin: Royal Family 'Don't Communicate Very Well'	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAGCsB5.html
2	3	0.161782	It's Not All About the Corgis - Here Are the Royal Family's Other Beloved Pets	We all know how much Queen Elizabeth loved her Corgis.	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AADmbCD.html
3	4	0.158303	Cutest photos of the royal Cambridge kids	See the best photos of Prince George, Princess Charlotte and Prince Louis!	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAAM6VA.html
4	5	0.157612	What Do Prince George & Princess Charlotte Know About Their Royal Roles?	Do Prince William and Kate Middleton's kids know about their royal roles?	lifestyle	lifestyleroyals	https://assets.msn.com/labs/mind/AAGeZKp.html

输出的新闻推荐结果如下图所示：

	rank	similarity_score	News ID	Title	Abstract
0	0	0.442929	N23937	6 gorgeous royal family heirlooms that Kate Mi...	Kate Middleton wears priceless heirlooms inclu...
1	1	0.372986	N60434	It's Not All About the Corgis - Here Are the R...	We all know how much Queen Elizabeth loved her...
2	2	0.372986	N51340	Queen Elizabeth's Cousin Says Royal Family 'Do...	Queen Elizabeth's Cousin: Royal Family 'Don't ...
3	3	0.270929	N1810	Where Do the Royals Reside? This Handy Guide W...	Even serious British royal family buffs might ...
4	4	0.239435	N34169	When royals lose their tempers, from the Queen...	See all the times the royal family have lost t...
5	5	0.153879	N40494	Cutest photos of the royal Cambridge kids	See the best photos of Prince George, Princess...
6	6	0.150135	N51736	2020 Ford Mustang Shelby GT500: 8 More GT500 E...	The more you know...
7	7	0.146020	N50187	Grover Norquist: Elizabeth Warren wants to rai...	Elizabeth Warren fashions herself a serious po...
8	8	0.129699	N8071	25 Photos of the Royal Family at Balmoral Cast...	The royal family has been visiting the Scottis...
9	9	0.093596	N14219	The rewriting of Democratic presidential campa...	Moderate Democrats are worried about one thing...

可以看到新闻推荐结果也都与皇室相关。

改进思路

仅仅使用tf-idf的新闻推荐算法的效果相比于当前成熟的新闻推荐模型比较一般，还可以使用其他的方法在该算法的基础上进行结合改进

1. 基于知识图谱的新闻推荐

   在MIND数据集中还提供了新闻的知识图谱嵌入等信息，利用知识图谱能更好地找到新闻之间的关联性，从而优化推荐效果。

2. 基于用户相似度的协同过滤

   当今时代人与人的联系愈加紧密，因此用户与用户之间的相似性也可以纳入考虑，当两个用户的浏览历史十分相似时，我们就可以利用协同过滤的想法，为一个用户推荐另一个用户观看过的新闻，实现更加智能的新闻推荐。

3. 动态的新闻推荐

   当前的新闻推荐是在新闻库中寻找新闻资料，但是在实际生活中，新闻每分每秒都在产生，可能有许多新出现的热点新闻是用户更感兴趣的。

参考资料

1. TF-IDF 原理：https://www.cnblogs.com/biyeymyhjob/archive/2012/07/17/2595249.html
2. 基于内容的新闻推荐算法：https://blog.csdn.net/qq_32690999/article/details/77434381
3. MIND数据集描述：https://github.com/msnews/msnews.github.io/blob/master/assets/doc/introduction.md
4. Tfidfrecommender：https://github.com/microsoft/recommenders/blob/master/examples/00_quick_start/tfidf_covid.ipynb