新闻聚合中的相似性分析和专题生成

新闻聚合是一种向用户提供新闻阅读服务的系统，它从多个新闻源中获取新闻，并按照一定的规则进行处理，整合，推荐以及分发。与传统报纸相比，新闻聚合服务能够根据用户的浏览记录和兴趣，从海量数据中为用户推荐其可能感兴趣的新闻。新闻聚合系统每天会从互联网上爬取数以万计的新闻，加上历史积累的数据，整个系统中会存储数以千万计的新闻数据。新闻聚合系统必须有一种行之有效的方法对这些海量数据进行高效处理。本文提出了一种基于相似哈希的方法，能够高效实现新闻抓取中的去重，相似性分析以及生成新闻专题。

1. 新闻聚合

在互联网时代，很多传统新闻行业已经在逐渐转型，通过互联网向用户提供新闻服务。但是这些网络服务只是简单地将纸质新闻数字化，并不能为用户提供个性化推荐。而已谷歌新闻为代表的新闻聚合服务，其本身并不生产新闻内容，

### 相似哈希

相似哈希算法[4]是由Charikar首次提出的，是局部哈希算法的一种并常被用作降维技术[5]。使用传统哈希算法，越相近的数据经过映射后差异越大。而相似哈希算法却与之不同，越相近数据被映射之后的差异越小。相似哈希的一个优点在于能够将处理对象的特征矢量映射成若干位(一般使用32，64或128bit)的一串数字哈希值。两个对象之间的相似性用其对应的若干位哈希值之间的距离来表示，一般使用汉明距离，距离越小则两者之间的相似性越高。

首先我们通过分词、去停用词、提取词干等方式来提取新闻特征。然后使用TF-IDF等方法来将加权后的新闻特征序列转化为一个高维稀疏矩阵。每则新闻被转化为一个稀疏的特征向量，相似哈希算法再将其映射成一个f位的数字哈希值。

### 新闻特征的稀疏矩阵表示

如果要对每一篇新闻进行相似哈希处理，首先要将其映射为一个特征向量。常用的方法是将新闻的文本进行分词，每一个词当做一个特征并用TF-IDF等方法将其变成一个加权的特征向量，重要使用的技术[6]即步骤如下：

#### 分词

分词是指将输入文本分割成单词、短语等其他有意义的元素。常用的分词方法可以采用按照文本格式（如空格）、常用短语分析、语义分析和机器学习等多种形式。

在我们的新闻聚合服务中，将新闻文本中的每一个单词当做一个特征，故使用正则表达式将其按照文本格式进行分词即可。

#### 去停用词

在一种语言中，都有很多出现频率非常高的词汇，如英语中的a/an/the/is等，这些频繁出现的词汇被称为停用词。停用词因为其几乎会在每一篇文本中高频出现，由信息论可知其能够提供的信息量很少，故对于文本的特征描述并没有什么用处。一般在信息检索、自然语言处理中，都普遍采用去除停用词这一方式来减少这些高频词汇的干染。

#### 提取词干

在英语等多种语言中，很多次都有其同意异形体，如单复数形式和前后缀。在新闻中，某些词因为上下文环境的不同会有不同的变体，这些同意异形体不利于文章的特征提取，所以一般采用提取词干的方式来将这些同意异形体规约到相同的形式。

#### TF-IDF

TF-IDF是当前在文本处理中较为流行的特征表示方式，它采用”bag of words”模型，将词频(TF)和逆文档频率(IDF)结合起来，既考虑当前文章的特征词分布，又考虑到了这些特征词在整个语料库中的频率。这样就能够使得我们的文本处理结果与我们的语料库所吻合，因为不同的文本处理任务，其语料库所形成的语义环境也是不一样的。

###### BoW模型

在文本分类中，最常见的是“bag of words”模型：输入特征向量中的每一个位置都对应于一个给定的单词。例如，单词“free”的出现可能是一个用于区分垃圾邮件的重要特征。BoW模型就是将文章中能够体现一篇文章的特征提取出来，并用这些特征来表示这篇文章。

###### TF

Term Frequency 指的是一个特定的单词出现在一篇文章的次数，在去除一些停用词之后，我们可以认为一篇文章中的词汇频率分布能够近似表示一篇文章的特征。比如我们来对比两篇关于体育赛事的新闻，前者报道了近期的一场足球赛事而后者的报道内容却是关于环法自行车赛。这两篇文章中肯定会出现一些相同的体育相关词汇和一些不同的具体词汇，但其频率分布应该呈现一定的差异。

###### DF

Document Frequency 是指某个特定单词在整个语料库的多少篇文章中出现过的次数。DF只计算某个单词在文章中出现与否，假设某个单词在某篇文章中出现了多次，在计算DF的时候也只能认为该单词在这篇文章中出现了1次。

###### TF-IDF

TF相对来说比DF更重要，因为DF是基于一个单词在一篇文章中出现与否的二进制数值，所以它并不能刻画一个单词在某一篇文章中的实际分布。比如，两个单词在某一篇文章中的TF分别是10和100，但是它们在这篇文章中的DF却都是1.

但是，TF同样有一些问题，就是他不能在整个语料库的基础上来判断某些词汇的重要性，比如英文中的'a'和'the'等词汇，在语料库的大部分文章中都出现了很多次，说明其对于将来要预测的文章来说并不是很重要的（因为从信息论的角度来看，我们已经知道这些词汇出现在这篇文章中的概率很多，我们所能得到的信息量是很少的）。

而TF-IDF的作用就是将特定词汇在整个语料库中某篇文章出现的概率和其在特定文章中概率分布结合了起来。

###### TF，DF，与TF-IDF的计算

1. TF=某个词在特定文章中出现的次数/该文章中的单词总数
2. TF-IDF=TF\*IDF

在经过上述步骤之后，整个语料库中的新闻被映射称为一个特征向量，每一个特征表示语料库中一个词汇。由于整个语料库中的词汇多达数万到数十万，而一篇新闻中只可能有这些所有特征中的一小部分。因此，我们可以稀疏矩阵来表示这些特征矢量，减少我们的系统内存，加快我们的运行时间。

### 特征降维

在上述的特征向量表示中，我们采用了“bag of words”模型来计算TF-IDF,这样只要是在整个语料库中出现的单词都会出现在我们的特征中，那么整个特征向量的维度可能达到数万到数十万维。一般来说，这些特征向量的维度比整个训练集中的文件数目要高出一个数量级。过高的维度不仅影响机器学习训练过程的效率，同时也会影响训练结果的精度。特征降维[7]就是通过一定的算法来减少输入的特征向量的维度，这是一个好的文本处理算法的基石。

#### 特征降维与训练效果的关系

特征降维对于大规模机器学习问题之所以重要，是因为其可以优化学习过程中的计算，存储以及将来对于已训练模型的每一次应用。更为重要的是，好的特征选择可以大幅度地提高机器学习算法的准确性。特征降维虽然降低了特征维数，但是却能提升性能，虽然这其中的原因目前各说纷纭，但这一事实却被广泛接受。

#### 特征降维的两类主要方法

降维技术（DRT，dimension reduction technology）有两类主要方法，特征变形(feature trans formation )和特征选择(feature selection)。

###### 特征变形

在特征变形中，原始的高维空间被投影到低维空间中，低维空间中的每一个维度都是原始高维空间的线性或者非线性组合。主要的技术有

1. Principal Components Analysis (PCA)
2. Factor Analysis, Projec- tion Pursuit
3. Latent Semantic Indexing (LSI)
4. Independent Component Analysis (ICA)
5. Random Projection(RP)

###### 特征选择

而特征选择则是从原始特征空间（矢量）中选择一些有用的特征，这就涉及到具体的文本处理任务和相关的特征重要性的评判标准。

具体的选择过程是选择某个评判标准，根据我们的文本处理任务（如分类），对特征向量中的每一个特征进行评测，并将评测结果用数值表示。本文采用chi-2评测标准来对上述特征向量进行选择，原特征向量为35000维左右，在选择了其10%最重要的特征之后，特征向量的维度变为3500左右。

### 相似哈希算法的具体实现

本文中的哈希算法的实现基于谷歌提出的相似文本去重[8]，由于我们采用了不同的特征表示方法并且调整了欧氏距离的门限，使得我们的系统能够找到相似但并不重复的相似新闻。实现过程主要分为两个部分，计算新闻的哈希值是基础部分，并且实现了一种高效的查询方法来快速地在数千万条哈希值中检索与特定哈希值相近的条目。

#### 计算相似哈希值

与传统的哈希算法不同，相似哈希能够将特征相似的对象映射成欧氏距离相近的哈希值。假设我们已经有了从文章中抽取出来的特征及其权重，我们可以用simhash算法来产生一个f位的哈希值（新闻的特征指纹）。相似哈希的计算过程如下：

1. 我们实例化一个f位的矢量V（可以是f位的数组），并将其每一位初始化为0。可以采用任一传统哈希算法(如MD5)将特征向量中的每一个特征映射成一个f位的哈希值。
2. 根据特征的f位哈希值来更新矢量V:如果该f位哈希值的第i位为1，在矢量v的第i位上加上该特征的权重，否则，减去该特征的权重。对特征向量中的每一个特征执行该步骤
3. 当所有的特征都被处理之后，矢量V的各个位置上可能是正值或者负值。如果为正值，则将该位设为1，如果为负，则将该位设为0。然后将该f位的矢量V转化为一个f位的数值，这就是该新闻的相似哈希值。

对每一篇新闻用上述算法进行处理之后就能得到相应的f位哈希值，我们仅需要储存该哈希值即可在后续过程中进行相似性判断。其中f的大小选择通常有32，64以及128这三种选择，通常要根据处理问题的规模进行判断。f越大，哈希值就越精确，但相应的所需存储空间也就越大。我们的新闻数目在1000万条左右，选择64bit即可。

#### 哈希值的存储与查询

我们在上一小节将每篇新闻转化为了f位的哈希值，如果我们要找到特定新闻的哈希值，只需用同样的步骤计算其相似哈希值，然后从所有已经计算出来的新闻哈希值中找到与特定新闻哈希值欧氏距离小于阈值的新闻即可。

但是我们的新闻数目在1000万条左右，如果要找特定新闻的哈希值，就需要将其与所有的1000万条新闻的哈希值相比较，这一过程所需的时间是我们所无法接受的。那么有没有什么办法可以快速进行查找呢？

我们可以发现，在1000万条新闻中可能有数篇到数百篇与特定新闻相似，而我们的新闻聚合服务可能只需要推荐数篇相似新闻即可。所以，我们完全可以牺牲一定的准确性来加快我们的探查时间。

如果两条新闻相似，其欧氏距离应该小于一定阈值，假设为d，那么这两条新闻的f位相似哈希值只有k(k<d)位不同，而其他位上都是相同的。我们可以将f位的哈希值分成若干部分，每一部分相同的哈希值存储在同一个存储块中。这样当我们要找特定新闻的相似新闻时，只需要按照simhash算法计算其f位的哈希值，并将f位其分成若干部分，然后从与每一部分相同的块中寻找可能相似的哈希值。通过这一方法，可以极大地减少所需探查的新闻数目。

那么我们应该将f位哈希值如何分块，来达到一个存储空间与探查时间上的平衡？假设我们采用f为64这一实现，我们思考以下几种分块方法：

1. 假设我们将64位分成8个部分，每个部分有8位。如果我们1000万条新闻（223），那么每一存储块中平均下来有223-28=32768条新闻，总的探查空间为8\*32768= 262144条新闻
2. 假设我们将64位分成5个部分，各部分的位数分别为13，13，13，13，12。平均情况下，每一个存储块中有223-213=1024条新闻，总的探查空间中有5\*1024=5120条新闻。
3. 假设我们将64位分成4个部分，每个部分16位。平均情况下，每一个存储块中有223-216= 128条新闻，总的探查空间中有4\*128=512条新闻。

我们可以看出，上述三种方案中，第二和第三种是可以接受的。第二种方案，存储空间是第三种方案的1.25倍，相应的查询准确度高于第三种方案。而其查询空间是第三种方案的10倍，相应的查询时间要长于第三种方案。总的来说，第二种方案的准确度要高一些，但其所需的计算资源（存储空间和计算时间）要多于第三种方案，所以要在准确度和计算资源之间做出一定的权衡。

#### 哈希值存储方案的实验比较。

我们对上一小节中的第二和第三种方案分别进行了实现，以下是我们的实验结果：

|  |  |  |
| --- | --- | --- |
| 位数 | 时间 | 准确率 |
| 13 | 0.0197474776505 | 0.78014680051 |
| 16 | 0.00364639523202 | 0.586880453038 |

其中时间表示查询一条特定新闻的相似新闻所需的平均时间，而准确率表示查询结果的召回率，也就是我们所查询到的相似新闻与该其实际的所有相似新闻的比值。

因为我们的相似新闻服务只需要为特定新闻找到数篇相似新闻即可，所以第三种方案是完全可以满足我们的需求的。

## 五、总结

本文介绍了新闻聚合服务的兴起和技术热点，并详细阐述了基于用户点击的新闻推荐和相似新闻这两个服务的具体实现。与一些理论上的机器学习方法不同，该文的工程实现能够满足实际的工程需求，实现简单，服务运行所需的计算资源较少，尤其是用相似哈希算法来实现的相似新闻服务，可以在极短时间内找到所需的相似新闻，而又不必对新加入的新闻做重复的处理操作。通过本文介绍的两种服务实现，我们可以构建用户体验较好的新闻聚合服务。

## 参考文献

1. Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66-72.
2. Zhao, Z. D., & Shang, M. S. (2010, January). User-based collaborative-filtering recommendation algorithms on hadoop. In Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on (pp. 478-481). IEEE.
3. Liu, J., Dolan, P., & Pedersen, E. R. (2010, February). Personalized news recommendation based on click behavior. In Proceedings of the 15th international conference on Intelligent user interfaces (pp. 31-40). ACM.
4. Charikar, M. S. (2002, May). Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (pp. 380-388). ACM.
5. Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. The Journal of machine learning research, 3, 1289-1305.
6. Lewis, D. D. (1992, February). Feature selection and feature extraction for text categorization. In Proceedings of the workshop on Speech and Natural Language (pp. 212-217). Association for Computational Linguistics.
7. Fodor, I. K. (2002). A survey of dimension reduction techniques.
8. Manku, G. S., Jain, A., & Das Sarma, A. (2007, May). Detecting near-duplicates for web crawling. In Proceedings of the 16th international conference on World Wide Web (pp. 141-150). ACM.