

# 基于乐纹特征和倒排索引的音乐检索系统

俞鹏飞 张新峰 王敏捷

(复旦大学计算机科学技术学院 上海 201203)

**摘要** 提出一个基于乐纹特征和倒排索引的音乐检索系统。该系统由预处理、特征提取、索引和精匹配四部分组成。通过两次基于动态阈值的筛选,选取频谱中最为稳定的点作为特征点,将特征作为关键词,采用倒排索引实现系统的初次查询。而精匹配则是对初次查询的优化和重排序,用优化后的编辑距离来计算两个特征序列的相似度。实验结果表明,提取的特征数据较小,系统具有较高的鲁棒性和查询准确率。

**关键词** 音乐检索 乐纹特征 倒排索引 鲁棒性 精匹配

中图分类号 TP3 文献标识码 A DOI:10.3969/j.issn.1000-386x.2014.10.010

## A MUSIC RETRIEVAL SYSTEM BASED ON AUDIO FINGERPRINT AND INVERTED INDEX

Yu Pengfei Zhang Xinfeng Wang Minjie

(School of Computer Science, Fudan University, Shanghai 201203, China)

**Abstract** This paper presents a music retrieval system, it is based on audio fingerprint and inverted index. The system consists of four parts: preprocessing, feature extraction, index, and precise matching. We select the most stable points in the spectrum as the feature points after two screening based on dynamic threshold, take the feature for keyword, and use inverted index to implement initial query of the system. And the precise matching is the optimisation and reordering on the initial query. We use the optimised Edit distance to calculate the similarity of the two feature sequences. Experimental results show that the feature data of audio features is very small and the system has high robustness and query precision.

**Keywords** Music retrieval Audio fingerprint Inverted index Robustness Precise matching

## 0 引言

随着网络 and 智能终端的迅速发展,人们对音频检索提出了较高的要求。基于文本描述的音频检索已经不能满足实际需求。而基于内容的音频检索方法,根据信息本身的特征参数而非外部属性对内容进行检索。因此,能更好地表示音频信息,从而使检索更加有效。

目前,音频特征已有很多,如 MFCC<sup>[4,12]</sup>、Chroma<sup>[2,3]</sup>、Fingerprint<sup>[1,7,10]</sup>等,MFCC 研究较早,它是基于人耳听觉特性的,但随着频率的提高,其计算精度随之下降,只适用于低频。文献[11]给出了一种基于小波包变换的音频特征,它可对全频率进行细化和分析,但时间复杂度较大。Chroma 是一种旋律特征,通过计算各频段的能量得到旋律特征,在哼唱检索<sup>[5,9]</sup>中使用较多,但其抗噪能力低于乐纹特征。乐纹特征从频谱图上提取特征,经哈希量化,具有较高的鲁棒性,是研究热点。

音乐检索作为音频检索<sup>[13]</sup>的一种,研究人员相对较多,包括哼唱检索和非哼唱检索。基于哼唱的音乐检索发展比较慢,它对查询片段的要求较高,且很难对系统进行评价。非哼唱音乐检索系统较多,包括 Shazam<sup>[1]</sup>、MusicDNS、Super Mbox 等,其中 Shazam、MusicDNS 和文献[8]提出的检索系统都使用了乐纹特征,可进行录音片段的查询。MusicDNS 检索系统没有建索

引,直接使用特征矩阵进行匹配,不适合大数据音乐检索,而 Shazam 和文献[8]的检索系统使用哈希进行查询,当数据规模较大时,哈希冲突会非常多,将会影响系统性能。同时这三种检索系统对于录音片段的查询,系统性能略有不足。本文特征改进了 Shazam 的乐纹特征,使其较鲁棒,且特征数据更小。检索算法则是通过两次检索实现,初次检索使用倒排索引,可用于大规模数据检索和分布式数据检索。精匹配实现二次检索,使用改进的编辑距离来计算两个特征序列的相似度。本文检索系统适用于海量音乐检索,且能够对录音歌曲进行有效的查询。

## 1 系统设计与实现

该系统主要包括预处理、特征提取、索引和精匹配四个部分。预处理主要完成音频信号的转换、重采样、滤波和信号增强;特征模块是对音频文件的表示,本文使用改进的乐纹特征,与其他乐纹特征<sup>[1,7,8]</sup>相比,本文特征数据较小,且更鲁棒;索引模块是根据歌曲库的特征建立倒排索引,当查询时,通过倒排索引获得所查特征的相关歌曲,并给出初次查询结果;精匹配在索

收稿日期:2013-04-24。国家自然科学基金项目(61071133,91024011);核高基重大专项(2010ZX01042-002-003-004);国家重点基础研究发展计划项目(2010CB731401)。俞鹏飞,硕士,主研领域:音频检索。张新峰,博士。王敏捷,博士。

引模块的基础上,结合音频特征间的时序关系,优化索引结果,同时获得查询片段在目标歌曲的具体位置。系统框架如图 1 所示。对于数据库中每首歌,通过预处理和特征提取,将特征保存于特征库中,用于建倒排索引和精匹配;对于查询片段,做相同的预处理和特征提出,其特征用于查询和精匹配。

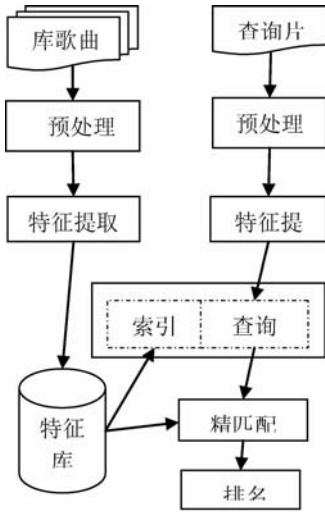


图 1 系统框架图

### 1.1 预处理

一般,查询片段含有大量噪音,特别是录音片段,随着录音设备和录音环境的不同,噪音变化较大。因此,需要对音频进行预处理,降低其噪音。通常,可以设计一个带通滤波器,只提取有效的音频频段。人声的频段在 50 Hz ~ 1 200 Hz,而乐器的频率在 10 kHz 以下,可以取 50 Hz ~ 10 kHz 或者更窄作为有效频段。此外,音频文件格式较多,采样频率、声道等有所不同,需要统一格式。根据最高有效频率和采样定理,采样频率为 20 kHz,一般取 16 kHz,同时转化成单声道音频文件。

### 1.2 特征提取

本文音频特征改进了 Shazam<sup>[1]</sup> 的乐纹特征,在文献[1]中,以极值点作为特征点,特征点非常多,导致特征数据较大,且很多特征抗噪能力较差。本文通过对这些特征点进行两次基于动态阈值的筛选,保留那些局部能量最大的点,这些点往往是频谱中最稳定的点,抗噪能力较强,从而大大减少特征点个数,同时也提高了特征的鲁棒性。

首先,对音频序列  $X = \{x_1, x_2, \dots, x_L\}$  进行分帧,帧之间有较高的重叠率,共分成  $M$  帧。接着,对每一帧进行快速傅里叶变换,取  $N$  个点,处理完所有帧后,将得到  $N$  维的频谱矩阵  $S_{N \times M}$ 。不同于文献[1],需要对频谱矩阵  $S$  进行平滑处理,使得后续的阈值变化较为平缓,计算公式如下:

$$S_{ij} = \log_{10}(\max(\text{abs}(S_{ij}), 1e^{-5})) \quad (1)$$

$$i = 1, 2, \dots, N \quad j = 1, 2, \dots, M$$

然后,在  $S$  中选取极大值点,即  $S(i, j) > S(i, j - 1)$  且  $S(i, j) > S(i, j + 1)$ ,作为特征点,并根据阈值来筛选特征点。用向量  $thresh$  表示各维阈值,通常取前几帧各维最大值来初始化各维阈值。顺序扫描所有特征点,若该点值大于对应维度的阈值,则保留该特征点,否则删除该特征点,同时按以下公式更新阈值向量  $thresh$  :

$$thresh_k = \begin{cases} thresh_k \times 0.97 & s(i, j) \leq thresh_k \\ \max(thresh_k, thresh_k \times e^{-\frac{(k-j)^2}{1000}}) & \text{其他} \end{cases} \quad (2)$$

$$k = 1, 2, \dots, N$$

从公式可以看出,若该点为特征点,则提高该维度及邻近维度阈值,否则,衰减各维度阈值。通过这次筛选,特征点的数量会下降 10 倍左右。从最后一个特征点开始,逆序扫描所有特征点,按相同原则筛选特征点和更新阈值。通过第二次筛选使得在某一较小区域,特征点的个数尽可能少,通常只有一个。图 2 是某段音频序列两次特征筛选的结果,充分说明了这点。



图 2 特征点筛选结果

最后,表示这些特征点,用一个点对来表示一个特征。对于每一个特征点,用它的邻近区域的每个特征点组成一个特征。很多时候,邻近区域内特征点较多,为了减少特征个数,选取与它最相邻的三个点组成三个特征,如图 3 所示。

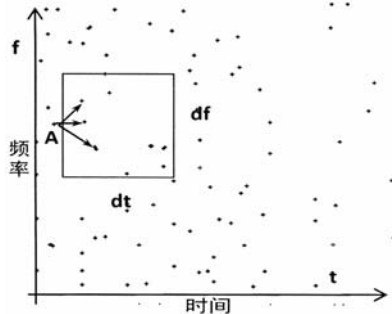


图 3 特征表示

图 3 中,点  $A(t, f)$  为要表示的特征点,矩形为它的邻近区域  $[t - dt/2 : t + dt/2, f + 1 : f + df]$ ,区域中最相邻的三个特征点与点  $A$  组成三个特征,即图中三个箭头。由于使用一个查询片段来进行检索,而非整首歌,对于时间维度,我们只需要两点的时差,且这两点是相邻的,它们的频率差比较小。所以用第一个点的频率  $F1$ ,第二点与第一个点的频率差  $\Delta F$  及它们的时差  $\Delta t$  来表示一个特征。为了便于后续检索,我们对特征进行了哈希计算,公式如下:

$$Feature = F1 \ll (fbits + tbits) + \Delta F \ll tbits + \Delta t \quad (3)$$

其中,用  $fbits$  位表示频率差,  $tbits$  位表示时差,  $Feature$  为特征值。当  $fbits$  或者  $tbits$  较小时,还需要进行取模运算。一个  $hash$  值便是一个乐纹特征,可以用 2 ~ 3 个字节来表示,而一首歌的特征个数在几千个,因此特征文件非常小。图 4 为一歌曲片段加噪前后的特征比较,纵轴表示加噪前后相同特征个数占总特征

个数的百分比,横轴为加入高斯随机噪的信噪比,特征为本文特征和 Shazam 的特征。可以看到,本文特征优于 Shazam 的特征;在信噪比 23 db 时,正确特征个数百分比在 60% 左右,还有较大的改进空间。

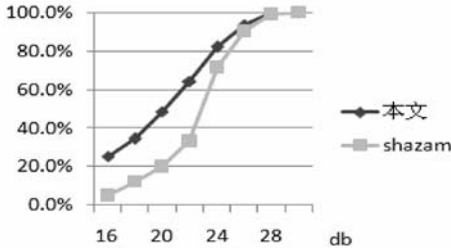


图 4 特征鲁棒性

由于每一帧的特征个数并非固定的,可以有零个或者多个,为了将特征序列和音频文件对应起来,需要在特征序列中,按一定的时间间隔,插入时间标记。这些时间标记不参与索引和精匹配,用于确定该标记前后的特征在音频文件中的具体位置。通常,时间间隔设为 1 秒,每个标记需要记录该标记在原音频中的具体秒数。为了统一格式,时间标记采用与特征相同的位数表示,且  $fbits + tbits$  位已经足够表示对应的秒数。因为每个特征的频率大于零,所以把 F1 为零时的哈希值作为时间标记,其数值为对应的秒数。

### 1.3 索引

基于乐纹的音乐检索系统通常采用哈希表来做查询,如文献[1,8]等,当数据库中歌曲增多时,哈希冲突也随之增多;当数据规模较大时,某些特征的冲突会非常多,影响其系统性能。本文采用倒排索引,它是一种文本检索技术,较为成熟。对于海量数据的查询,也有相应的查询系统,如 Lucene<sup>[14]</sup>等,可以直接使用,并且具有较快的查询速度。

在建立索引时,关键词的选择非常重要,可以把一个特征或者连续的几个特征组成关键词。本系统用一个特征作为一个关键词,且不考虑时间标记。因为查询片段带有较多噪音,每个特征都有一定的差错率,连续几个特征将会扩大差错率,很可能导致关键词错误。根据数据库中特征,建立倒排索引,每一个特征作为一个索引项,记录其相关歌曲及其权重。对于一个查询片段,通过倒排索引获得查询片段中每个特征与库中各歌曲的权重,并计算权重和,作为查询片段与各歌曲的相似度。根据权重和,对各歌曲排序,其结果作为索引返回。

由于查询片段较为随机,查询片段的特征很可能是目标歌曲的偏僻特征。当查询片段较短时,偏僻特征对检索影响非常大,它使得目标歌曲排名靠后。因此,在建索引时,应把所有特征的权重设为相等,也就是不存在偏僻特征,但这样大大降低了目标歌曲与其他歌曲的区分度。通过对歌曲进行分段,以段为单位建立索引,能有效地提高它们之间的区分度。用 1 万首歌,根据段长 30 秒,1 分钟,2 分钟和不分段分别建立索引,用相同的 130 个查询片段进行分析,目标歌曲出现在前  $k$  项的百分比如表 1 所示。分段后,返回结果明显优于分段前,随着分段长度的减小,结果越来越好;取分段时间为 1 分钟较合理,短于 1 分钟时,结果改进不大。

表 1 分段后索引结果

	Top - 10	Top - 30	Top - 50
0.5 min	86.2%	93.8%	93.8%

续表 1

	Top - 10	Top - 30	Top - 50
1 min	87.7%	91.5%	93.1%
2 min	76.9%	84.6%	86.9%
不分段	73.8%	80.8%	80.8%

### 1.4 精匹配

倒排索引忽略了特征之间的时序关系,在噪音较大或者查询片段较短的情况下,导致目标歌曲很难与其相似歌曲区分,这就需要进一步计算,找到目标歌曲。另外,索引对歌曲进行了分段处理,需要还原成歌曲名排序,并去除重复曲目。通过匹配算法来精确计算查询特征序列与候选特征序列的相似度,从而根据相似度来优化索引返回的结果。

一般音频序列的匹配采用动态时间归整(DTW)<sup>[6]</sup>,但特征已经过哈希计算,各维度权重差别非常大,并不适用。本文采用两个序列的编辑距离作为它们的相似度,即计算一个序列经过多少次修改转化为另一个序列,值越小,越相似。但在较大噪音下,查询序列和原序列出入比较大,这就要求匹配时,有一定的容错能力,因此在计算编辑距离时,把哈希值错一位的两个特征看作是正确匹配。假设两个特征的哈希值为  $A, B$ ,那么它们之间的距离如式(4)所示,当  $A, B$  相等或者错一位,  $D$  为 0, 否则为 1。

$$D = \min((A \neq B) \& ((A \neq B) - 1), 1) \quad (4)$$

对于匹配算法,时间复杂度是较为关键的问题,本文算法时间复杂度为  $O(k \times n \times m)$ ,其中  $n, m$  分别表示两个比较序列的长度,  $k$  为索引返回的候选序列个数。通常,候选序列较多且长度较长,系统响应较慢。因此,需要对算法进行优化,减少候选序列个数及缩短其长度。我们知道,初次查询是按相同特征个数多少进行排序的。假定排序后,相同特征个数  $num_i$ , 如果存在一“拐点”,使得:

$$\sum_{i=1}^K num_i > \sum_{i=K+1}^{3 \times K} num_i \quad 3 \times K \leq N \quad (5)$$

$$num_i \geq num_j \quad i < j$$

则认为目标歌曲就在这前  $K$  个候选序列中;  $K$  越大,则目标歌曲在候选序列中的概率越低;如果  $K = 1$ ,表明结果已经较为理想,不需要重新排名;一般情况下,  $K$  小于 10;当  $K$  不存在时,表明无法找到目标歌曲,无需计算。

对于前  $K$  个候选序列,先对这些序列进行分段(其长度略长于查询片段,较高重叠率),找出最为相似的几个片段,它们与查询序列具有最多的相同特征个数。系统中选取 3 个片段,过多反而影响匹配速度。将这 3 个片段进行修改后的编辑距离计算,把最小编辑距离的片段作为与查询序列最相似的片段,取最小编辑距离作为该候选序列的相似度,并记录该片段中第一个时间标记,作为查询片段在该歌曲中的具体位置。最后,我们根据相似度进行排序,得到最终的检索排名及在相应歌曲中的具体位置,如果某首歌在排名中出现多条记录,只保留第一条记录。

## 2 实验与分析

在实验中,我们一共下载了 25 000 多首歌,包括中文、英文、韩文、日文等歌曲,时长一般在 3~6 分钟,个别较长。输入是一个歌曲片段,时长一般在 5~20 s。在实际过程中,查询片段往往是带有噪音或者失真的。我们通过多种方法生成查询片段,如加入高斯白噪声、录音、重采样、音频格式转换等。实验表

明,系统对重采样,音频格式转换后的查询片段具有非常好的性能,同时,在单任务下,系统响应是实时的。文中主要对加入高斯白噪声和录音的查询片段作分析。为了便于比较,查询片段的文件名以目标歌曲名作为前缀,以此来判断目标歌曲是否在检索结果中。通过播放查询片段和检索返回的歌曲片段,来判断返回的具体位置是否准确,由于具有一定的主观性,实验中不进行描述。

**定义 1** 查询准确率:每次查询返回前  $k$  项,若目标歌曲在前  $k$  项中,则此次查询成功,否则失败。查询准确率定义为,在  $N$  次查询中,查询成功的次数占总查询次数的百分比。

首先,对本文特征进行测试。我们将本文特征与文献[1, 8]使用的特征进行对比,采用本文检索系统。查询片段为 120 个录音片段,时间长度在 10~30 s,结果如表 2 所示(文献[1]的特征为乐纹 1,文献[8]的特征为乐纹 2,本文特征为乐纹 3)。从表中看出,本文特征具有较高的鲁棒性。

表 2 特征比较

	命中	Top - 1	Top - 10	Top - 30
乐纹 1	个数	62	69	76
	准确率(%)	0.52	0.58	0.63
乐纹 2	个数	91	93	96
	准确率(%)	75.8	77.4	80.0
乐纹 3	个数	110	111	113
	准确率(%)	91.7	92.5	94.2

其次,对精匹配效果的测试。查询片段采用录音歌曲,且长度在 10 s 左右,共 100 个片段,初次查询返回前 50 个片段。从图 5 中看出,如果目标歌曲在初次查询 top - 10 中,精匹配能有效地找到目标歌曲;如果目标歌曲排在较后面,那么精匹配效果并不明显。

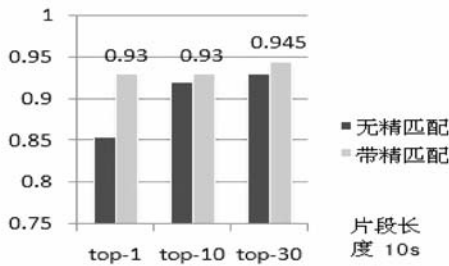


图 5 精匹配效果

然后,我们对带有高斯白噪声的查询片段进行测试。这些查询片段是由库中歌曲片段加入高斯随机噪声产生的,片段长度在 5~15 s 左右,共 100 个片段,返回 top - 10 结果如图 6 所示。我们可以看到:查询片段时间长度,对系统性能影响不大;当信噪比达到 20 dB 以上,查询准确率达 90% 以上。

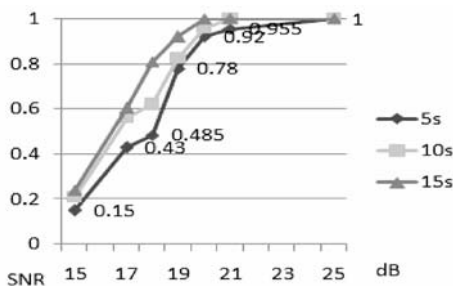


图 6 带高斯白噪声查询准确率

最后,我们对录音歌曲进行测试,用录音笔在不同环境下录制 200 个查询片段,来自于 60 首歌,且每首歌都已在数据库中,结果如图 7 所示。

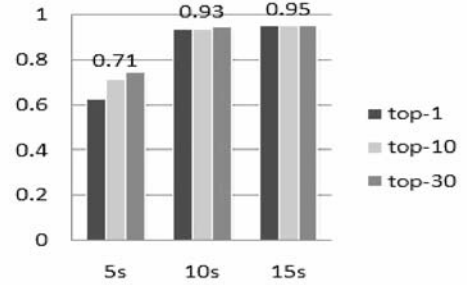


图 7 录音歌曲查询准确率

从图中可以看出:对于时间长度在 5 s 左右的查询片段,查询准确率较低,而且精匹配效果差;对于长度在 10 s 左右的查询片段,其查询准确率较为理想,且基本上第一首歌即为目标歌曲;对于超过 15 s 的查询片段,准确率提高已不明显。因此,对于录音查询片段,时间长度在 10 s 以上,其具有较高的查询准确率,同时具有较好的精匹配效果。

### 3 结 语

本文的特征方法不但具有较高的鲁棒性,而且数据量非常小。不同于其他音乐检索系统,我们把特征转为关键词,使用倒排索引实现初次检索,同时在系统中加入了精匹配,由于其结合了特征之间的时序关系,使得排序结果更理想。虽然精匹配具有较大的时间复杂度,但通过多步优化使系统具有较快的响应时间。总体上看,该系统性能良好,可适应于录音片段的查询。整个系统除了预处理和特征提取,其他过程都是按文本的方式处理的,可以把系统整合到文本检索系统中。

### 参 考 文 献

- [1] Wang. An industrial strength audio search algorithm[C]//ISMIR Proceedings, 2003.
- [2] Barrington L, Chan A B, Lanckriet G. Modeling Music as a Dynamic Texture[J]. IEEE Transactions on audio, speech, and language processing, 2010, 18(3): 602 - 612.
- [3] Wang Chungche, JyhShing Roger Jang, Wang Wennan. An Improved Query by Singing/Humming System Using Melody and Lyrics Information[C]//ISMIR, 2010.
- [4] Mizanur Rahman, Md Babul Islam. Performance Evaluation of MLPC and MFCC for HMM based Noisy Speech Recognition[C]//ICCIT, 2010.
- [5] Cristina de la Bandera, Ana M Barbancho, Lorenzo J Tardon, et al. Humming method for content-based music information retrieval[C]//ISMIR Proceedings, 2011.
- [6] Muller M. Information Retrieval for Music and Motion[M]. Springer, 2007.
- [7] Liu Chihchin, Chang Pofeng. An efficient audio fingerprint design for MP3 music[C]//Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia, 2011.
- [8] Jaap Haitsma, Ton Kalker. A Highly Robust Audio Fingerprinting System[C]//ISMIR Proceedings, 2002.
- [9] Kichul Kim, Kang Ryoung Park, Sung-Joo Park, et al. Robust query-by-singing/humming system against background noise environments[J]. IEEE Transactions on Consumer Electronics, 2011, 57(2): 720 - 725.

最终发现在所有训练集合中,可以进行链接的  $t$  总数为 24 758 个,平均每一条微博有 3.08 个  $n$ -gram 可以进行链接。在选择进行预测的三个特征中,比较重要的特征是  $n$ -gram 的  $P(t)$  和  $IDF(t)$ 。

表 2 LR 的结果

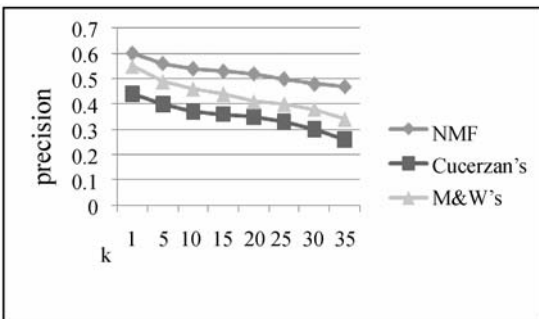
Precision	Recall	F-measure
0.847 6	0.843 5	0.845 5

利用 NMF 方法得到的初步结果如表 3 所示,例子中给出了 2 个概念的语义相关度最大的前 5 个概念。

表 3 NMF 分解得到的前 5 个最相关的概念向量

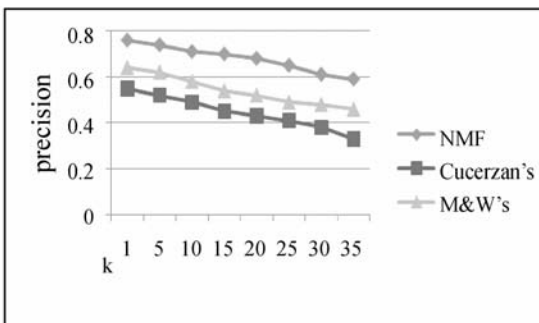
$C_1^{(5)}$	$C_2^{(5)}$
United States	President of the United States
United States of America	United States Senator
US	African-American
USA	White House
Americans	2008 presidential election
American	Barack Obama

本次实验中设置矩阵的秩  $r = 50$ ,迭代停止误差为  $tot = 0.000001$ ,迭代次数为 8 000。针对不同计算方法得到的结果集合,作为比较选取了不同的结果集合范围。不基于上下文的语义概念扩展方法与其它两种方法得到的准确率如图 1 所示。

图 1 给定秩  $r$  在不同范围  $K$  的结果正确率(不基于上下文)

对于不同的范围  $K$ ,基于 NMF 的方法在准确率上均比其它两种方法有较大提升。从曲线的趋势可以看出,基于 NMF 的方法梯度下降较慢,具有比较好的扩展性。由于选择的语料只是维基百科的一个子集,在以后的工作中可以选择更全面的语料进行深入研究。

在不基于上下文的基础上,增加了利用上下文信息扩展语义概念的结果如图 2 所示。

图 2 给定秩  $r$  在不同  $k$  的正确率(基于上下文)

可以看到,在原有方法基础上再结合上下文语义,三种方法的实验结果都获得了比较大提升,证明了基于上下文语义概念扩展方法的有效性。

## 4 结 语

本文提出了一种基于 NMF 的方法来计算维基百科中锚文本的语义相似度,并在此基础上为短文本增加语义知识。基于部分语料的实验表明该方法具有较好的性能。而该方法不仅可以寻找到短文本中有意义的概念实体,还可以增加语义相似的同类知识,因此可以扩展到规则过滤、短文本聚类等领域。接下来还将从以下两个方面对本文的方法进行改进:一方面增大实验数据集合,对该方法在大数据集合上的性能进行深入验证;另一方面通过一种自动化寻找最优的分解秩的方法对 NMF 进行改进,提高程序的运行效率和精度。

## 参 考 文 献

- [1] Sitaram Asur, Bernardo A. Huberman. Predicting the Future With Social Media[R]. ACM, 2010.
- [2] Takeshi Sakaki, Makoto Okazaki, Yutaka Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors [C]// WWW, 2010.
- [3] A Mathes. Folksonomies-cooperative classification and communication through shared metadata [R]. Computer Mediated Communication, 2004.
- [4] S Banerjee, K Ramanathan, A Gupta. Clustering Short Texts using Wikipedia [C]//Proceedings of the SIGIR, ACM, 2007:787-788.
- [5] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data [C]//EMNLP conference, 2006, 9-16.
- [6] Paolo Ferragina. TAGME: On-the-fly Annotation of Short Text Fragments [R]. CIKM, 2010.
- [7] Xianpei Han, Le Sun, Jun Zhao. Collective Entity Linking in Web Text: A Graph-Based Method [R]. SIGIR, 2011.
- [8] D Milne, I H Witten. Learning to link with Wikipedia [R]. CIKM, 2008.
- [9] F Abel, Qi Gao, Geert-Jan Houben, et al. Semantic Enrichment of Twitter Posts for User Profile Construction on the social web [R]. ESWC, 2011.
- [10] Patrik et. Non-negative Matrix Factorization with Sparseness Constraints [J]. Journal of Machine Learning Research, 2004(5):1457-1469.

## (上接第 48 页)

- [10] 张敏, 欧阳建权, 李泽洲, 等. 一种快速的特定音频指纹提取方法 [J]. 计算机工程, 2010, 36(2): 211-213.
- [11] 魏国华, 郑继明. 基于局域判别基的音频信号特征提取方法 [J]. 计算机应用与软件, 2009, 26(11): 84-87.
- [12] 俞玉莲, 郭世杰. 基于支持向量机的多类音频分类 [J]. 计算机应用与软件, 2010, 26(11): 221-223.
- [13] 李森, 孙荣坤, 韩纪庆, 等. 基于 p-稳定分布局部敏感哈希地址的鲁棒音频检索方法 [J]. 信号处理, 2012, 28(3): 367-375.
- [14] 姜鑫, 余平. 基于 Lucene 的音视频资源检索系统的研究与实现 [J]. 计算机应用与软件, 2011, 28(11): 245-248.