密级：_____

浙江大学

硕 士 学 位 论 文

论文题目 基于移动用户地理信息的音乐推荐研究

作者姓名 夏飞

指导教师 陈刚教授

寿黎但副教授 陈珂副教授

学科(专业) 计算机应用技术

所在学院 计算机学院

提交日期 2013-01

A Dissertation Submitted to Zhejiang University for the Degree of Master of Engineering

TITLE: Situational Music Recommendation In Mobile Environment

Author: Xia Fei

Supervisor: Chen Gang

Shou Lidan Chen Ke

Subject: Computer Application Technology

College: Computer Science and Technology

Submitted Date: 2013-01

摘要

在本文中研究了一种新型的音乐推荐方法，即基于移动用户地理信息的音乐推荐。和传统的音乐推荐系统不同，本文所实现的音乐推荐系统要根据移动用户的地理信息来进行音乐推荐。传统的音乐推荐中不会考虑用户的位置，但在移动环境中，用户所处的位置随时会发生改变，周围的环境会影响到用户当前的心理状态，继而影响到用户对音乐的感受以及期待，比如当一个人面朝大海或处于喧嚣的都市之内时收听同一首歌的感受必然不同，因此推荐给用户的音乐也应随之改变。

本文设计并实现了叫做“Pictune”的音乐推荐系统，它能利用网上大量的照片数据和其他网络多媒体数据向处在任何位置的用户推荐音乐。我们给出了 Pictune 详细的系统结构以及音乐推荐流程分析。其中本文主要研究并阐述了其中的两个难点，第一个难点是任意英文单词相似度的计算，单词相似度计算的高效性和有效性对 Pictune 的实现来说必不可少，我们在前人工作的基础上提出了三种算法来解决这个问题，分别是基于 Lucene 索引的方法、常驻内存的方法以及多核并行计算的方法；另一个难点是推荐算法的设计和实现，本文详细讨论了基于生成模型的照片数据建模方法，在经典的 LDA 模型中引入了空间信息，给出了基于 EM 算法的模型参数估计的方法，并基于此模型设计了高效的推荐算法。

本文最后给出的大量实验表明上述算法不仅有效而且高效，实现的 Pictune 音乐推荐系统具有非常实用的价值。

关键词： 音乐推荐，语义分析，空间索引，主题模型

Abstract

In this paper we study a new type of music recommendation, namely situational music recommendation in mobile environment. Different from traditional music recommendation systems, we will take user's current geographical ambience into consideration in situational music recommendation. Mobile user's location is constantly changing as well as his geographical ambience. User's feeling and expectation of music is significantly influenced by his inner-state which in turn is greatly affected by his geographical ambience, thus we have to make different recommendations according to user's geographical ambience.

In this paper we propose a new music recommendation prototype named Pictune. Pictune utilizes location-based image retrieval and other Web services to recommend music for users virtually located at any position. We analyze the architecture and recommendation process of Pictune in depth. We tackle mainly two difficult problems in this paper, the first is arbitrary English word similarity calculation, the effectiveness and efficiency of word similarity calculation is very important to pictune, we propose three different methods to solve this problem, namely Lucene-Base method, Memory-Resident method and MultiCore-Based method; Another problem is the design and implementation of recommendation algorithm, we devise a variant generative topic model for the photo data taking photos' GPS coordinates into consideration, this model is based on the famous LDA topic model. We use Expectation Maximization algorithm for parameter estimation, and design an efficient recommendation algorithm.

Extensive experiments on real data sets confirm the effectiveness and efficiency of our proposed algorithms and demonstrate the practical value of Pictune.

Keywords： Music Recommendation, Semantic Analysis, Spatial Index, Topic Modeling

目录

图目录

表目录

第1章 绪论

1.1 课题背景与研究内容

近年来音乐推荐系统(Music Recommendation System)不仅在学术界得到了深入探讨和研究，在工业界也已经有了诸多的设计和实现，比如 Last.fm[1]、Pandora[2]、豆瓣 FM[3]都是比较成功的音乐推荐网站应用。传统的推荐系统所需要解决的问题大致可以分为两个子问题。一是预测判断问题，即评估某个用户（User）U 对一个物品（Item）I 的喜好程度；二是根据预测出的喜好分数高低推荐出最合适的 N 个物品给用户。其中预测判断是推荐系统中的最核心的问题，因为一旦能将物品按照预测出的分数排序，推荐任务就可以简单地转化为提取前 N 个物品。

当前的音乐推荐系统都假设用户是静止的，但在现实生活中用户很多情况下都是处在户外或是处在运动的状态下。随着这几年移动终端以及 LBS（Location Based Service）服务的普及，移动用户的数量呈现出指数级增长的趋势。有数据表明大量的音乐推荐请求来自于移动终端而非主机。这就产生了问题，传统的音乐推荐系统依赖于用户的信息档案（Profile）来进行音乐推荐，但是用户的信息档案往往是静态不变的，不论用户处在地球上的任何地方推荐给她的音乐都是一成不变的。但不论是生活的经验还是相关研究均表明我们人在聆听音乐时的感受与用户当时的心理状态有关，而用户的心理状态强烈地受到她周围环境的影响，试想当一个人面朝大海或处于喧嚣的都市之内时收听同一首歌的感受必然不同。也就是说并不存在全局最优的歌曲(即在地点 A 最合适的歌曲在地点 B 不一定最合适)，在音乐推荐的过程中应该考虑进用户的环境因素。

与传统的音乐推荐服务不同，移动用户通过移动终端能提供的信息非常多，比如用户的当前位置、用户朝向等等。普通的音乐推荐系统只能通过协同过滤（Collaborative-Filtering）(它假设相似的用户喜欢的歌曲也是相似的，因此给用

[1] www.last.fm
[2] www.pandora.com
[3] douban.fm

户推荐的是其他类似的用户喜欢的歌曲）和基于音乐内容分析（Content-Based Analysis）等方法来静态地推荐音乐。而我们系统的目的是根据用户所处环境和时间的不同实时地推荐最合适的音乐给用户。

本文主要研究根据用户所处的地理位置所提供的信息向用户推荐音乐。现在大部分的移动设备都能提供包括 GPS 坐标、时间在内的信息，根据这些信息本文要解决两个大问题。一是如何根据 GPS 坐标（表达为经纬度形式）和时间等信息获得和抽象出用户所处环境的信息；二是如何根据第一步抽象出的环境信息进行音乐推荐。下面举一个例子来说明本文的应用场景：

在图（1.1）中用户从地点 A（浙江大学）移动到地点 B（雷峰塔）的过程中，周围的环境一直在改变，用户看到的环境对他的影响应该体现于推荐的歌曲中：在地点 A 的时候系统可以推荐和浙江大学有关的歌曲给用户，比如浙大校歌；在地点 B 的时候系统可以推荐和雷峰塔相关的歌曲给用户，比如千年等一回。

图 1.1 一个应用场景例子

和传统的音乐推荐方法不同，本文所实现的 Pictune 音乐推荐系统要根据移

动用户的地理信息来进行音乐推荐。传统的音乐推荐中不会考虑用户的位置，但在移动环境中，用户所处的位置随时会发生改变，周围的环境会影响到用户当前的心理状态，继而影响到用户对音乐的感受，因此推荐给用户的音乐也应随之改变。Pictune 系统首先要解决的问题就是如何表达用户所在的地理环境对用户的影响。要知道地理环境对用户的影响，首先要能知道该地理环境的主要特征，比如某个地方是否是度假旅游区或者是否临海等等信息。要知道这些特征，需要描述性的信息，而且这些描述信息必须同时具有地理坐标或者具有清晰易辨的描述地理位置的内容，否则我们无法确认这些描述信息所描述的地点。本文的做法是使用网络上大量的具有标签、评论和 GPS 坐标等元数据的照片作为描述性文档，因为它们同时拥有 GPS 坐标以及可以作为描述性文档的标签、评论等信息，根据这些内容我们可以建立模型计算出用户所在位置的特征。根据这些特征我们可以确认哪些音乐最符合这些特征，进而推荐这些音乐给用户。如何根据照片元数据信息建立这个模型以及基于该模型的推荐算法的设计是本文的其中一个重要研究内容，我们会在第六章中详细阐述。

另外在计算音乐和用户所在地理位置的特征是否匹配的时候，我们也必须知道音乐所表达的信息，和照片的处理方式类似，本文使用音乐的歌词、标签等元数据信息作为音乐的描述性文档。但是因为歌曲的歌词、标签的用词习惯和照片用户对照片打标签的用词习惯差异很大，所以即使对照片数据的模型建立得比较完善，也很难根据这个模型来计算音乐和用户所在位置特征的匹配程度，这是因为模型都是基于描述性的文字建立的，如果我们对英文的照片集建立了一个主题抽取的模型，但是我们拿这个模型去计算匹配程度最高的中文歌曲，当然不可能得到好的结果。这里的问题类似，如果不能解决这两个数据集合间用词的差异度的话，推荐效果就很难保证。本文采用的方法是将其中一个数据集合的单词转换到另一个数据集合的里的单词，而转换的过程需要计算任意单词之间的相似度，因为即使需要转换也应该转换成和原始单词意思最相近的单词。因此任意单词之间相似度的计算是本文的另一个重要的研究内容。

1.2 本文工作及贡献

传统的基于协同过滤和音乐内容分析的方法都不能处理移动用户的音乐推荐请求，因为它们都没有考虑用户的地理信息：基于用户的协同过滤算法（User-Based Collaborative Filtering）通过计算用户的相似度来寻找和当前用户类似的 Top-K 个用户，并根据他们的喜好来进行推荐，基于物品的协同过滤算法（Item-Based Collaborative Filtering）则根据物品之间的相似度进行推荐；基于音乐内容分析的方法通过音乐底层听觉信息诸如韵律、音高和类型、乐器等高层信息来推荐相似的歌曲。

解决移动用户的音乐推荐问题需要将用户的环境信息融合进推荐算法中。第一个需要解决的问题就是环境信息的表达。直观上应该用结构化的数据来表达地理信息，但结构化的数据虽然方便查询处理，但不能很好地说明地理特点。比如如果我们用 R-Tree 等空间索引结构来索引地理位置，我们只能快速地查找某个地理位置或范围，但无法得知该位置或范围的特征，因此在我们的模型中既融合进了地理信息也考虑了文本内容。

本文创新之处列举如下：

（1） 提出了一种非常实用的音乐推荐应用：基于移动用户地理信息的音乐推荐

（2） 设计并实现了一种可行的基于地理信息的音乐推荐系统原型

（3） 设计了一种高效易用的任意英文单词相似度计算方法

（4） 设计了一种高效的音乐推荐算法

（5） 通过实验验证了整个推荐系统的有效性和高效性

1.3 本文组织

本文总共分为八章，每一章的内容如下：

第一章介绍了本文的研究背景和研究内容，阐述了基于用户地理信息的音乐推荐系统的新颖性和必要性。

第二章回顾了本文的相关工作，包括空间索引、传统音乐推荐系统、单词相

似度的计算等内容。

第三章给出了本文中涉及到的数据模型和要解决的问题定义。

第四章介绍了本论文实现的基于用户地理信息的 Pictune 音乐推荐系统的整体结构和音乐推荐流程。

第五章详细介绍了任意单词对之间相似度的计算方法，分别设计了基于 Lucene 索引的算法、常驻内存的算法和基于多核并行计算的算法。

第六章详细阐述了地理主题的模型设计以及基于地理主题模型的音乐推荐算法。

第七章给出了实验结果及实验分析。

第八章总结了本文的工作，并展望了未来的工作。

1.4 本章小结

本章首先介绍了本文研究的课题背景和研究内容，然后说明了本文的创新之处，最后给出了全文的组织结构。

第2章 相关工作

本章将回顾和基于地理信息的音乐推荐系统的相关工作，为后续工作的展开提供参考。本文主要涉及空间数据库领域的索引机制与查询方法，以及信息检索领域针对海量多媒体对象的检索推荐技术以及音乐推荐技术。2.1 节首先描述空间数据库领域的索引和查询技术；2.2 节介绍传统的音乐推荐方法；2.3 节讨论一些较新的和地理位置有关的音乐推荐方法;2.4 节是关于单词相似度的计算技术；2.5 节介绍了文档主题建模相关方法；最后在 2.6 节总结了本章内容。

2.1 空间索引和查询

和传统的音乐推荐系统不同，本文提出的音乐推荐系统是为了解决移动端的音乐推荐问题，所以在推荐的过程中必须要考虑用户实时的位置信息，该位置信息是由诸如手机等移动端设备发送给服务器的，在我们的推荐系统中将会根据用户的经纬度坐标进行计算。另外在本系统中将会访问海量的具有 GPS 信息的图片，我们的系统要求能对这些数据进行高效的索引和快速的查询，所以传统空间数据库领域里的技术将在我们的系统中发挥重要的作用。

在空间数据库领域，为了快速、有效地访问海量空间数据，专家学者提出了大量的空间索引方法，常见的索引方法包括网格（Grid）、四叉树[1]（Quad-Tree）、R 树索引[2]、R+树索引[3]、R*树索引[4]、K-D-B 树索引、Hilbert 曲线索引[5]。在此基础上，更提出了各种各具特色的查询及其解决方案，如近邻查询、K 近邻查询、连续近邻查询、反向近邻查询、最远邻居查询、skyline 查询。这些空间索引通常以层次型的结构组织空间对象，从而支持高效的空间查询。以被广泛采用的 R 树为例，空间上位置相近的数据点被聚类到 MBR（最小包围矩形），这些 MBR 又根据空间局部性递归的进行聚类，直到到达根节点。

在这些查询中和我们的系统最相关的是范围查询和 KNN（K nearest neighbor）最近邻查询。当用户的位置表达为一个 GPS 点时系统进行 KNN 查询，若用户的位置表达为一个矩形或者一个圆形时系统进行范围查询。文章提出了基于 R-Tree

的矩形区间的范围查询，圆形区间的范围查询也一样，只不过在计算数据点是否被包围的时候要查看某个数据点距离圆心的距离是否小于半径。文章[6]给出了一种基于 R-tree 的 KNN 计算方法：在自顶向下的查询过程中，维护一个优先队列，优先队列里储存了在递归查询中碰到的 MBR 或者数据点，并且以和查询点距离由小到大排序，MBR 和查询点间的距离就是查询点到 MBR 某条边的最短距离，这样递归查询下去直到优先队列的队头出现了数据点，则该数据点必然是查询点的最近数据点。该方法的优点是不需要事先固定 K，可以增量式查询，而且是 IO 最优化的。

近年来随着移动客户端越来越普及，静态的 KNN 查询已经不能满足要求，所以学术界提出了针对移动用户的 KNN 查询算法。在移动 KNN 问题中，假设移动用户会持续查询离自己最近的 K 个物体。文献[7]给出了四种算法，分别是固定上限的算法、懒惰搜索算法、预取查询算法和双缓存算法。后三种算法都是为了极小化查询计算代价的递进改良算法，它们依赖于历史查询结果和预取查询结果。文献[8]结合了静态的空间划分数据结构 Voronoi-Diagram[9]和动态的安全领域结构（Safe-Region 安全领域是指不论用户在其中如何移动都不会改变查询结果的区域）提出了一种称为 V^*-Diagram 的动态增量算法。传统的基于安全领域的移动 KNN 算法在计算安全领域时只考虑数据点而不考虑查询点的位置，该文章在计算安全领域时会同时考虑两者。所以 V^*-Diagram 算法消耗少得多的 IO 和计算资源。

2.2 传统的音乐推荐方法

2.2.1 人口统计过滤

人口统计过滤可以用来识别喜欢某种物品的用户类型。比如可以用来学习出哪些类型的人喜欢某个歌手。这种方法根据用户的个人信息、地理信息、和心理信息将用户档案（User Profile）进行分类。早期的 Grundy system [10]就是基于人口统计过滤的推荐系统。Grundy system 通过从交互的对话框获得的信息来向用户推荐书籍。这个方法的主要问题是推荐给具有类似用户档案的的用户的物品都是相同的。另一个严重的问题是用户档案的产生需要用户手动输入。另一些复杂点

的系统通过用户的主页、博客等资源获得关于用户的信息，然后使用文本聚类的方法将用户分类。虽然该方法的缺点较多，但却是最简单的方法。

2.2.2 基于协同过滤的推荐

协同过滤的方法通过研究历史的用户-物品关系来预测用户对某些物品的喜好程度，也就是说用户提供反馈给系统，然后系统基于他人给出的这些反馈信息为某个用户做出推测。早期的一个基于协同过滤的音乐推荐系统是 Ringo[11]。协同过滤方法通过建立一个表达用户对物品喜好的矩阵来工作。矩阵的每一行代表一个用户信息，而每一列代表一个物品。R_{u_i,i_j}代表用户u_i对物品i_j的喜好程度。图（2.1）就是这样的一个矩阵：

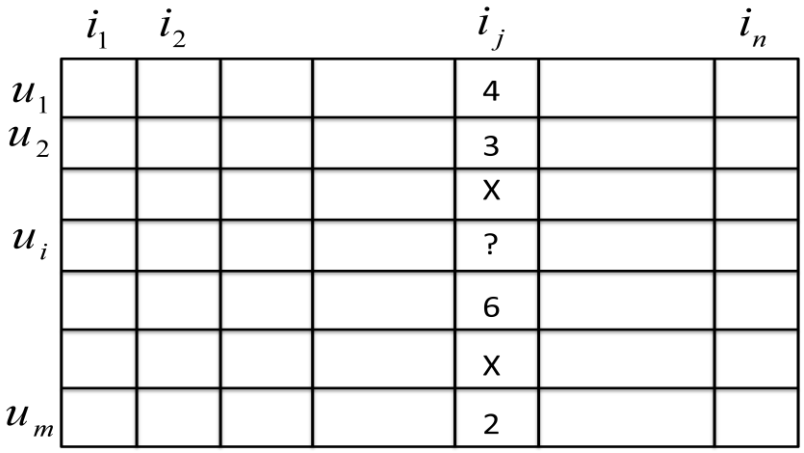

图 2.1 用户喜好矩阵

2.2.2.1 基于用户(User-Based)的协同过滤

预测的用户 u 对物品 i 的喜好值$P_{u,i}$可以由和该用户类似的用户对物品 i 的打分的平均值算出。下式给出了这样预测出来的喜好值。$\overline{R_u}$是用户 u 的平均喜好值，$R_{u,i}$代表用户 u 对物品 i 的喜好值。但是为了预测出$P_{u,i}$，系统必须预先知道和 u 相似的用户集，$v \in Neighbor(u)$，而且要知道他们有多相似，即值$sim(u,v)$，还有这个相似用户集的大小 k。最常见的计算用户相似度的方法是 Pearson Correlation 和 Cosine Similarity。

$$P_{u,i} = \overline{R_u} + \frac{\sum_{v \in Neighbor\ (u)}^{k} sim(u,v)(R_{v,i} - \overline{R_v})}{\sum_{v \in Neighbor\ (u)}^{k} sim(u,v)}$$

2.2.2.2 基于物品(Item-Based)的协同过滤

基于物品的协同过滤计算的是物品之间的相似度而不是用户间的相似度。该方法首先查询出该用户已经打过分的物品集，以及物品集中物品和预测目标物品之间的相似度。图（2.2）展示了不同户打过分的相同物品集。这个例子中显示了物品i_j和物品i_k之间的相似度，而且在计算中只考虑了u_2和u_i而没有u_{m-1}，这是因为他没有同时为两个物品打分。

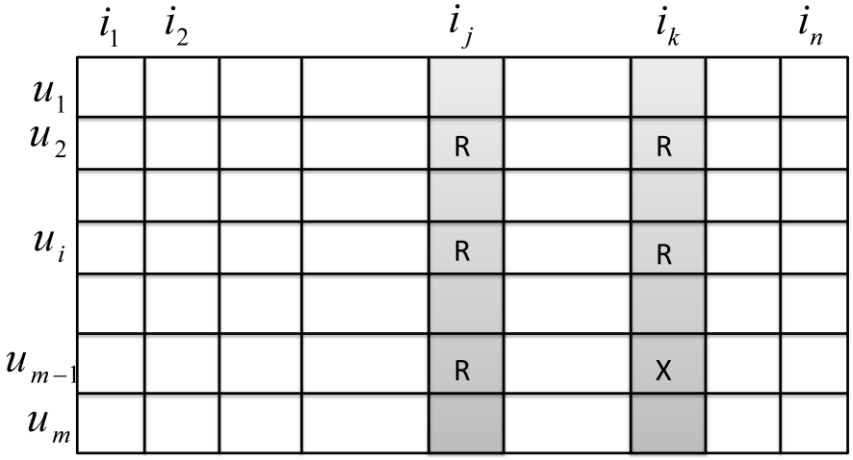

图 2.2 基于物品的协同过滤例子

物品之间的相似度同样可以使用 Cosine Similarity、Pearson Correlation、Adjusted Cosine Similarity 或者条件概率P(j|i)。如果用 U 代表同时为物品 i 和 j 打过分的用户集，$R_{u,i}$代表用户 u 对物品 i 的打分，下式给出了 Cosine Similarity 的计算公式：

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|i\| * \|j\|} = \frac{\sum_{u \in U} R_{u,i} R_{u,j}}{\sqrt{\sum_{u \in U} R_{u,i}^2} \sqrt{\sum_{u \in U} R_{u,j}^2}}$$

Cosine Similarity 没有考虑不同用户打分的分布区间问题，所以计算出来的物品间相似度是有问题的。Adjusted Cosine Similarity 对此作出了改进，下式给出该相似度度量的公式，其中$\overline{R_u}$是用户 u 的平均打分：

$$sim(i,j)=\frac{\sum_{u\in U}(R_{u,i}-\overline{R_u})(R_{u,j}-\overline{R_u})}{\sqrt{\sum_{u\in U}(R_{u,i}-\overline{R_u})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\overline{R_u})^2}}$$

Pearson Correlation 计算出的两个物品之间的值代表了物品之间的相关联性。下式定义了 Pearson Correlation，其中$\overline{R_i}$代表了第 i 个物品的平均得分：

$$sim(i,j)=\frac{Cov(i,j)}{\sigma_i\sigma_j}\frac{\sum_{u\in U}(R_{u,i}-\overline{R_i})(R_{u,j}-\overline{R_j})}{\sqrt{\sum_{u\in U}(R_{u,i}-\overline{R_i})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\overline{R_j})^2}}$$

假设已经计算出了和用户打过分的并且和物品 i 最相似的物品集$S^k(i;u)$，包括 k 个物品。那么预测出的用户对物品的打分是一个用户打分的一个加权平均值，下式给出了预测用户 u 对物品 i 的打分的公式：

$$P_{u,i}=\frac{\sum_{j\in S^k(i;u)}sim(i,j)R_{u,j}}{\sum_{j\in S^k(i;u)}sim(i,j)}$$

虽然协同过滤方法是广泛采用的一种推荐方法，但不论是哪种协同过滤方法都有着比较明显的缺点：

（1） 高维数据特征和数据稀疏性。系统中存在海量的用户和物品集，所以用户-物品矩阵是一个巨型矩阵，但是用户对物品的打分非常稀疏，很难有效地找到相似的用户/物品集。

（2） 对于一些比较非典型的用户，他们喜欢的物品和大部分用户的都不同，为他们找到用户、物品近邻更加困难，推荐的效果更不可靠。

（3） 冷启动问题。对于基于用户的协同过滤来说由于新用户的打分数据非常少，因此对新用户的推荐更加困难。同样，对于基于物品的协同过滤，新加入的物品只有少量的用户对它们打过分。

（4） 协同过滤算法只考虑了用户的反馈（表现为打分、买卖、下载等等社交类信息），没有考虑任何和物品本身相关的信息，因此常常推荐的是最受欢迎的而不是最合适的物品。

2.2.3 基于内容的推荐

在基于内容的推荐方法中，系统首先收集描述物品的信息，然后根据用户的

喜好信息推荐物品。这个方法不依赖于其他用户对物品的打分。提取物品信息的方法既可以是自动的（通过程序分析物品的内容并提取相应的特征）也可是手动的（由领域专家逐一提供），也可以使用社区用户对物品打上的标签（比如 Last.fm 用户对歌曲打上的标签）。

2.2.4 基于上下文的推荐

基于上下文的推荐方法和基于内容的方法不同，使用的不是物品的内容而是和物品相关联的上下文信息进行计算和推荐。比如对于垃圾邮件检测，基于内容的方法是分析邮件的文字内容、声音内容，甚至视频内容的特征；基于上下文的方法分析的是 SMTP 连接的上下文信息来推断邮件是否是垃圾邮件。常用的上下文方法有两种，网络信息挖掘（Web Mining）和社会性标签（Social Tagging），两者均能用来计算物品、用户间的相似度并提供有效地推荐结果。网络信息挖掘主要以分析网络上的内容信息作为基础；而社会性标签主要通过社区用户的打标签方式挖掘有用的信息。

2.2.4.1 网络信息挖掘方法

文献[12]中认为有三种类型的网络挖掘模式：分别是内容挖掘，结构挖掘，用途挖掘。

（1） 网络内容挖掘：包括文本、超文本、超文本标记和多媒体挖掘。同过这些挖掘的内容的分析，可以得出物品的相似度。该类别包括比如舆论提取（语义分析），博客分析，挖掘分析顾客评价，主题识别等等。

（2） 网络结构挖掘：主要侧重于连接分析（出链和入链）。也就是网络的拓扑结构的分析（类似 PageRank 和 Hits）。

（3） 网络用途挖掘：使用的是关于会话日志（Session Logs）的信息进行挖掘。这类信息可以用于导出用户的习惯和喜好信息，做出连接预测和基于会话日志共生状态的物品相似度计算。

结合这三种方法，推荐系统可以推导出物品间的相似度（比如出现在相同页面里的物品、在同一会话中被访问过的物品等等）。而且能基于用户和网络内容

的交互对用户建模。如果关于网络内容的信息是以文本的形式呈现的话，经典的信息检索领域的方法可以用来为这些内容建立模型和特征，比如空间向量模型可以被用来为物品和用户档案建模。物品描述（使用 bag-of-words 模型）和用户档案之间可以用 cosine 距离计算相似度。

一个经典的权重函数是$w(i,j)$是 TF/IDF。TF（Term Frequency）表示词频，IDF（Inverse Document Frequency）表示逆文档频率[13]。一个文档中的某个单词的 TF 表达了这个单词在该文档中的重要性。下面是 TF 的定义，其中n_i是该单词在相应文档中出现的次数，分母是相应文档的长度。

$$TF = \frac{n_i}{\sum_k n_k}$$

IDF 表达了单词在整个文档集中特殊程度，其中$|D|$是文档集中文档的数量，分母是包含了该单词的文档数目：

$$IDF = \log \frac{|D|}{|(d_i \supset t_i)|}$$

最后在 TF-IDF 权重计算中单词j在文档t中的权重等于：

$$w_{t,j} = TF \cdot IDF$$

另一个有用的计算物品相似度的度量函数是 PMI（Pointwise Mutual Information）PMI 通过计算两个物品同时出现的程度来得出它们之间的相似度。PMI 量化了两者联合概率分布和它们各自概率分布之间的差异度（假设独立）。

$$PMI(i,j) = \log \frac{p(i,j)}{p(i)p(j)}$$

本文的系统中使用了 TF-IDF 权重计算方式。

2.2.4.2 社会性标签方法

社会性标签主要目的在于对网络内容用标签进行标注解释。标签（Tag）是可以自由使用的文字、单词，并不一定约束于某一个既定的语言。

当用户对物品打标签时，我们得到的是一个元组（Tuple）<user, item, tag>。有了这些海量元组我们可以得到几个有用的矩阵：

（1） User-Tag 矩阵。$U_{i,j}$记录了用户 i 使用标签 j 的次数。使用 User-Tag 矩

阵可以得出用户的档案信息（比如为用户生成标签云表达他的兴趣爱好），或者计算两个用户爱好的相似度。

（2） Item-Tag 矩阵。$I_{i,j}$表示物品 i 被打上标签 j 的次数。该矩阵包含了物品的上下文描述信息。用此矩阵可以计算物品的相似度。

（3） User-Item 矩阵。$R_{i,j}$表示用户 i 给物品 j 打过标签。此矩阵可以用于经典的协同过滤算法。

总之，物品相似度和用户相似度可以从矩阵 I、U 得出。而且在此过程中可以使用 SVD（Singular Value Decomposition）方法和 NMF（Non-Negative Matrix Factorization）方法进行降维操作（如果碰到了数据稀疏的问题）。

只要物品（用户）相似度被计算出来了，User-Item 矩阵或者是用户档案都可以被用来为用户进行推荐预测。文章[14]提出一个使用上述三个矩阵推荐网页的框架。文章[15]使用 Item-Tag 矩阵来提高推荐结果的精确性，比单用协同过滤方法得出的结果精确性要高。文献[16]使用潜在语义分析（Latent Semantic Analysis）和在已降维空间中的 cosine 距离来计算和可视化呈现艺术家之间的相似度（使用从 Last.fm 音乐网站获得的标签）。

2.3 和地理位置有关的音乐推荐方法

在以往的研究中几乎都没有考虑用户的位置信息。文献[17]提出了一种为知名地点（POI）推荐音乐的方法。作者请了 32 个志愿者给 POI 和歌曲打上了标签，然后通过 Jaccard、Cosine 等等相似度标准为 POI 计算最相似的歌曲。该工作具有几个显著的缺点：它的方法不具有可扩展性，POI 和歌曲的标签都是依靠人工打上的，在数据量很大的情况下根本不可行；用户在漫游的时候其最近点的位置变化非常快，推荐的歌曲相应地也会频繁切换，造成剧烈抖动，作者并没有考虑这点；用户的最近点并不能很好地代表用户所处的环境，对用户影响最大的应该是用户所见范围内的物体；用户的最近点不一定适合进行音乐推荐，比如公共厕所。

文献[18]研究了如何从图片和歌曲的内容推测心情状况的问题。方法是，首先根据多媒体内容提取不同的特征，然后根据机器学习的方法，在心情模型（Mood

Model）的基础上进行学习，然后再对新的图片和歌曲分类。心情模型主要分为两类，一类主要是以一种方法列举形容词和名词；另一类是多维度模型。歌曲的特征主要是基于音色和节奏[19]，而图片的特征主要包括亮度、颜色、饱和度、对比度等[20]。依靠机器学习进行心情提取的缺点是训练集不易取得，而且效果较差。

此外在音乐推荐系统中，音乐的标签是非常重要的资源。文献[21]调研并对比了五种不同收集音乐标签的方法，分为：由音乐专家完成的音乐标注问卷，比如 Pandora 网站的“Music Genome”计划；社交网站用户打的标签，比如 Last.fm 网站；音乐标注游戏；网页挖掘；利用机器学习方法自动标注。

综上，上述已有研究主要集中在基于分析图片、音乐的内容来进行音乐推荐，还没有文献考虑本项目提出的结合用户所在地理信息进行实时音乐推荐的问题。

2.4 单词相似度计算

本文中的音乐推荐过程会同时访问 Flickr 照片集中的词汇和音乐中的词汇，但是两者之间的用词差别比较大，需要预先将其中一种词汇集往另一词汇集转换，这个过程需要高效的单词相似度计算函数。详细内容请参见第 6 章。本节列举一些单词相似度计算的相关工作。

2.4.1 潜在语义分析方法（LSA, Latent Semantic Anslysis）

LSA 方法可以说是[22]单词空间里面最为流行的方法了。该方法最为核心的部分是奇异值分解（Singular Value Decomposition，SVD）。奇异值分解把一个高维空间的矩阵转换成一个低维空间的矩阵，使得均方误差最小化（Least Mean Square Error）。

LSA 分析的目的是得出单词在给定的文档中的真正含义，也就是潜在语义。具体地说，就是经过 SVD 降维把文档集都用映射到一个低维空间，比如对于一个具有 3000 文档的文档集，LSA 可以使用 100 维度的空间将单词和文档都表示到该空间。降维是 LSA 中最关键的一部，通过降维，文档中的“噪声”（无关信息）都被除去了，语义结构呈现了出来。

在计算单词相似度的时候，我们首先把两个单词映射到该空间，把两个单词

都表示为该空间中的向量，然后用向量的相似度计算公式求得它们之间的相似度作为两个单词之间的单词相似度。

2.4.2 WordNet::Similarity

Wordnet::Similarity[23]是一个完全免费的软件包，它可以计算两个概念（concept）或者单词意思（word senses）之间的语义相似度（semantic similarity）和关联性（relatedness）。它分别提供了六种相似度的度量方法和三种关联性的度量方法，这些方法都是基于 WordNet[4]词汇数据库的。官网提供了这些方法对应的 Perl 语言模块。这些方法的输入是两个概念，输出是一个代表两个概念相似度或者关联性高低的数值。

概念间的相似度和关联性计算都是基于 WordNet 中 is-a 的层次结构。比如 automobile 和 boat 间的相似度要高于它和 tree 之间的相似度，这是因为 automobile 和 boat 在 is-a 层次结构中有一个公共祖先 vehicle。

WordNet 中将名词和动词都按照 is-a 关系组织了起来。在 WordNet 2.0 中名词对应了 9 个层次结构共包含了 8 万个概念，动词对应了 554 个层次结构，共 13500 个概念。

但是在 WordNet 里面 is-a 层次结构并不能跨越词性，所以 Wordnet::Similarity 只能用作计算名词对或者动词对之间的相似度和关联性。虽然 WordNet 里面包含了形容词和副词，但是它们并没有按照 is-a 关系组织，所以不能计算形容词对和副词对的相似度和关联性。

2.4.3 Disco 单词分布式相似度计算

这篇文章说明了单词相似度的计算应该考虑单词之间的共生特性，共生特性可以从句子的语法结构特征中提取：大量单词的组合搭配说明了特定单词组之间具有某种依存关系，也就说明了单词之间的关联性。Disco[24]在前人工作的基础上提出了语法结构的一个比较精细的替代品，他们用单词在滑动窗口中的距离来近似代替单词之间的共生语法特征，提出了一种非常实用的单词相似度计算方法，

[4] http://wordnet.princeton.edu/

而且可以计算任意单词之间的相似度。

但是这篇文章中的方法不具有可扩展性，单词间相似度的计算效率非常低下，本文中的第五章会在此基础上提出更为高效的单词相似度计算算法。

2.5 主题建模（Topic Modeling）

隐含狄利克雷分配（Latent Dirichlet Allocation）[25]是一个离散数据集的生成模型，比如文档集，LDA 是一个三层的贝叶斯模型。在 LDA 里面，每一个文档被看作由各种不同的主题（Topic）生成的，这一点和 pLSA（Probabilistic Latent Semantic Analysis）一样，只不过 LDA 里面假设主题的分布还包含了一个一个狄利克雷先验量（Dirichlet Prior）。

LDA 中假设每个文档的生成过程如下所示：

（1） 按照泊松分布抽样出文档的长度$N \sim Poisson(\xi)$。

（2） 根据狄利克雷分布抽样出参数$\theta \sim Dirchlet(\alpha)$。

（3） 对文档中 N 个单词中的每一个w_n：

从多项式分布抽样出w_n对应的主题$z_n \sim Multinomial(\theta)$。

从多项式分布$p(w_n|z_n,\beta)$中抽样出$w_n \sim p(w_n|z_n,\beta)$。

LDA 对应的图模型（Graphical Model）如图（2.3）所示。

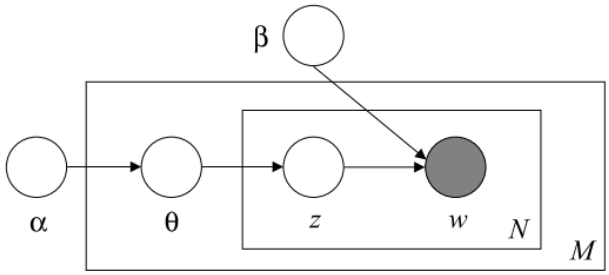

图 2.3 LDA 的图模型

LDA 中使用 EM 算法来估计模型参数值。在本文中的文档具有 GPS 坐标信息（Flickr 照片），所以不能看作纯文档，因此本文不能直接使用 LDA 中的生成模型，在第六章中我们会详细说明如何加入一个中间层来解决这个问题。

2.6 本章小结

本节列举了和本文相关的一些工作，包括空间索引结构、传统的音乐推荐方法、和地理信息相关的音乐推荐方法、单词相似度计算以及文档主题建模。

第3章 数据模型和问题定义

本章将详细阐述基于移动用户地理信息的 Pictune 音乐推荐系统的问题定义和数据模型。首先会在 3.1 节中对本文所涉及到的数据进行定义。然后在 3.2 节对本文涉及的推荐问题进行总体描述。最后在 3.3 节对本章的内容进行小结。

3.1 数据模型定义

定义 3.1 GPS 关联文档：是一种具有 GPS 坐标信息的文本文档。形式化的表示，一个 GPS 关联文档包含了一个单词集合$\mathbf{w}_d$，其中所有的单词均来自一个词汇表 V。$l_d = (x_d, y_d)$是该文档的 GPS 坐标，即x_d是经度，y_d是纬度。举个例子，一个 GPS 关联文档可以是一个 Twitter 上的一个 tweet，其中的文档内容$\mathbf{w}_d$就是该 tweet 里包含的文字信息，而l_d是发布该 tweet 时移动端的 GPS 坐标。另外在本文中提到的文档均不考虑其文字间的语法信息和结构信息，即对于文档使用 bag-of-words 模型处理。

在本文中将用从 Flickr 网站抓取的照片集来作为一个地区的描述信息，但是我们只需要 Flickr 网站中标注有 GPS 坐标的照片，只有这种照片对我们有用，形式化的来说，每张具有 GPS 坐标的 Flickr 照片可以表示如下：

定义 3.2 Flickr 照片：每张 Flickr 照片可以表示为一个元组$(id, l_d, \mathbf{w}_d)$，其中 id 是该照片在 Flickr 网站的唯一标识符，为一个 32 位整数，l_d是拍摄该张照片的移动客户端的 GPS 坐标，$\mathbf{w}_d$是表示该张照片对应的文本内容，包括：用户打上的标签、照片标题、评论信息等等。比如一张拍摄浙江大学宿舍的取名为 laundry 照片的 id 是 7278285106，标签为（china, hangzhou, Zhejiang University, campus, building, balcony, students housing, clothes）。在本文中用到的照片集合即为 P。

在本文所涉及的环境中用户一直处于变化的过程，不仅用户的位置一直变化，用户的心情也会随着环境的变化而产生相应的变化。用户的推荐请求通过用户所持的移动端提供给服务器端，服务器端根据这些信息计算出合适的歌曲返回给用户。形式化地，本文所涉及的移动用户请求可以定义如下：

定义 3.3 移动用户请求：每个用户请求可表达为一个元组$((u_x, u_y, k, r, t)$。其中u_x, u_y分别是用户的当前所处位置的 GPS 坐标，其中k值域和r值域不会同时存在，它们的用处将在接下去的文中进行说明。t代表用户发出请求的时间戳。

服务器端在收到移动用户的请求之后会为用户生成一个上下文信息，该上下文信息包含了描述用户所处位置环境的文档内容，形式化地移动用户上下文信息定义如下：

定义 3.4 移动用户上下文：上下文信息是一个照片集合P_u。集合P_u是根据定义 3.3 中的移动用户请求由服务器端计算出来的。在定义 3.3 中的移动用户请求中如果用户设定了k值，服务器会将查询出的和(u_x, u_y)在地理位置上最近的 k 张照片作为集合P_u；但如果用户设定了r域（r既可以表示矩形区域也可以表示圆心区域，由用户选择）服务器就会将该区域包含的所有照片当作用户上下文信息。

移动用户上下文信息表达了 3.1 节中描述的用户档案信息。直观来说在音乐推荐中要能根据用户所处不同环境做出最合适的推荐就必须要能高效地描述和记录用户所处地理位置的信息。移动用户上下文中包含的集合P_u中的照片正好描述了用户所在位置信息，这是因为每张照片里均有用户打上的标签和评论等信息，从这些文本信息中我们能提取出有效的地理主题信息：

定义 3.5 地理主题（Geographical Topic）： 每个地理主题是一个在地理空间中表达一致而且有意义的主题，和这个主题相关的地理位置在空间中具有临近性。也就是说在空间中出现相近的单词往往能组成一个主题。

定义 3.6 移动用户地理主题（Mobile Geographical Topic）： 表示根据移动用户上下文信息计算得出的地理主题。

上面只描述了和地理位置有关的信息的表达方式，为了进行有效的音乐推荐还需要对音乐进行建模，在本文中不考虑音乐的内容信息，即不分析音乐的内容。我们把描述音乐的信息称为音乐文档，定义如下：

定义 3.7 音乐文档（Music Document）： 对于音乐集合 M 中的每首音乐 m 用 bag-of-words 模型描述为一个文档$\mathbf{m}_d$，文档中的单词来自音乐网站中用户对该首音乐打上的标签以及音乐对应的分词后的歌词。

有了移动用户主题和音乐文档的定义，下面可以定义出两者之间的相似度，可用于音乐推荐中的度量：

定义 3.8 移动用户主题和音乐主题相似度（Similarity）计算函数 $\mathrm{sim}(u_{topic}, \mathbf{m}_d)$：可以使用若干方法计算两者的相似度，所得值越高表示相似度越高，匹配程度越强。本文中的推荐算法使用相似度高低推荐音乐。相应的相似度计算函数请参考第六章。

3.2 基于地理位置的音乐推荐

以 3.2 节中定义的若干数据模型作为基础，下面给出基于地理位置的音乐推荐问题的定义：

定义 3.9 基于移动用户地理信息的音乐推荐（Situational Music Recommendation）： 在移动的环境中，在给定移动用户请求和上下文信息的条件下，根据移动用户地理主题 u_{topic} 和音乐文档信息计算出最合适的音乐推荐给用户。计算的结果是一个音乐集合 M'（其中 $|M'|$ 是系统参数表示返回的音乐数量），满足对于任何 $m' \in M'$ 和对于任何 $m \in M$ 有 $\mathrm{sim}(u_{topic}, \mathbf{m}'_d) \geq \mathrm{sim}(u_{topic}, \mathbf{m}_d)$，而且 M' 中的音乐是根据 $\mathrm{sim}(u_{topic}, \mathbf{m}'_d)$ 降序排列的。

如前所述，单词之间的相似度也会极大地影响推荐系统的效果。由于本文研究的是英文的音乐推荐系统，Flickr 照片中的数据来源是英文标签、标题、评论，而歌曲集合也是英文歌曲。所以本文之后提及的单词相似度均是指英文单词之间的相似度。形式化地，可以定义如下的英文单词相似度：

定义 3.10 英文单词相似度（Word Similarity）：给定英文单词集合 V，和任意两个单词 $w_1 \in V$，$w_2 \in V$，w_1 和 w_2 之间的单词相似度是一个浮点值 $s \in [0,1]$，s 的值越大说明两者相似度越高，否则越低。

在后面的内容中凡是提及单词相似度均是指英文单词相似度。

3.3 本章小结

本章首先介绍了 Pictune 系统中所需要解决的问题，然后形式化地对本文涉

及到的所有数据进行了形式化定义，最后对 Pictune 系统中研究的基于地理信息的音乐推荐算法和单词相似度计算做出了定义。

第4章 系统结构和推荐流程

在本章中，我们将总体描述已经实现的 Pictune 推荐系统的大致结构以及音乐推荐流程。其中最主要的单词相似度计算模块和推荐算法模块将分别在第五章和第六章详细阐述。

4.1 系统结构

整个系统分为后台服务器和前端页面展示（系统运行效果图请参见第七章中的Demo展示部分），结构为B/S结构，以下是服务器端和页面前端所负责的任务：

（1） 后台服务器负责：

数据的抓取（图片、音乐、歌词等等），数据的存储，各种索引的建立和维护，运行时音乐推荐的计算等等。

（2） 前端负责：

页面展示，处理用户选择、输入，传递数据给服务器端，实时下载和显示图片，播放音乐等等。

如图（4.1）所示为整个系统的结构，具体说明如下：

（1） 用户通过页面内嵌的 Google Map 选择出查询区间，由页面前端计算出经纬度范围，另外用户还可以出入当前的心情信息，然后用户点击开始按钮之后前端将上述信息通过 AJAX 方法发送给服务器。

（2） 服务器接收到用户请求，通过已建立的图片索引系统查询出在用户所选范围内的所有照片信息。

（3） 基于用户所输入的心情信息对图片的标签进行一定的情感增强（Emotion Enhancement）。

（4） 通过查询引擎和音乐索引系统计算出最匹配的歌曲，在此过程中要使用任意单词间相似度计算的方法。

（5） 通过 AJAX 方法返回结果给浏览器端。

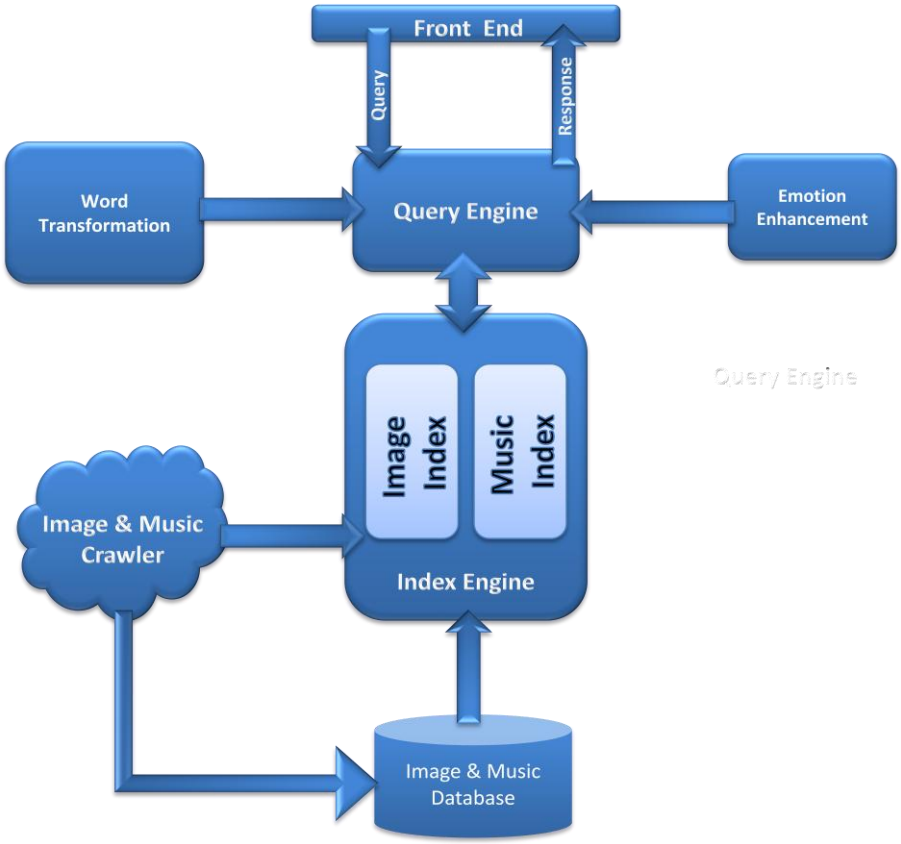

图 4.1 Pictune 系统结构

下面详细描述每个模块的功能：

4.1.1 服务器和浏览器前端通信模块

由于服务器端和浏览器端之间需要异步地发送和接收数据，所以必须使用 AJAX 方法，本项目选择 DWR (Direct Web Remoting)。DWR 是一个用于改善 web 页面与 Java 类交互的远程服务器端 Ajax 开源框架，可以帮助开发人员开发包含 AJAX 技术的网站。它可以允许在浏览器里的代码使用运行在 WEB 服务器上的 JAVA 函数，就像它就在浏览器里一样。

4.1.2 数据抓取模块

本文中设计的数据比较多，可以分为以下 3 个部分：

4.1.2.1 照片元数据抓取模块

为了得到地理位置的描述信息我们需要大量的具有 GPS 坐标的照片。由于

Flickr 网站有着海量的照片数据，所以照片的抓取是一件相当棘手的事情，需要考虑各种因素进行加速。照片数据的抓取细节请参见第七章“实验结果”。

4.1.2.2 音乐元数据抓取模块

同样为了得到音乐的描述信息我们需要大量的音乐歌词、曲名、歌手名等元数据信心。音乐元数据的抓取细节请参见第七章“实验结果”。

4.1.2.3 音乐抓取模块

推荐的最后需要传输推荐的音乐内容给用户，所以也需要大量的音乐内容数据。由于音乐内容数据的大小以 MB/首为单位，而且数据抓取还有音乐版权的问题，所以音乐抓取任务也比较棘手。音乐的抓取细节请参见第七章“实验结果”。

4.1.3 索引模块

1. 照片索引：

采用 R*-Tree 索引，每个数据节点包含了照片的 id 号并以照片的 GPS 坐标作为键值。

2. 音乐索引：

音乐元数据以及歌词按照顺序方式直接存放于内存中，因为这些数据量并不大。音乐数据存放于磁盘中，通过流数据形式从磁盘取出然后传输给浏览器进行播放。

4.1.4 单词转换模块

假设音乐中出现的所有单词的集合是 W，本模块的作用是将照片中出现的每个单词 x 转换成 W 中和 x 单词相似度最高的单词。这一过程中会大量用到单词相似度计算，单词相似度计算的方法详见第五章“单词相似度计算”。

4.1.5 标签分词模块

在第七章中描述的 Flickr 照片抓取方法中使用的 Flickr 网站的 API flickr.photos.search 函数有一个 bug，它返回的照片的标签集合中超过两个单词的标签会被连接在一起，比如标签若是 world of beauty 的话 flickr.com 返回的就是

worldofbeauty，现在的分词方法是基于一个单词库在每个标签中查找出现的单词，算法是 AC 自动机(Aho-Corasick String Matching Algorithm)[26]，单词库使用是英文维基百科词库，但是这种分词算法具有明显的缺点：经过分词后本来不存在的单词也会出现了，比如 mississippi 经过分词后会变为两个单词 miss 和 mississippi。因此只对过长的单词尝试分词，而且拆分出的单词长度大于某个阈值才留下。

4.1.6 音乐查询模块

详细内容请参见第六章“推荐算法”。

4.1.7 标签云生成模块

在推荐模块处理完成之后计算得出 TF-IDF 值最大的 Top-K 个标签，为这 Top-K 个标签计算标签云的算法描述如下：

While 不能在画布上放下所有标签：

通过二分法缩小每个标签所占据空间找到最合适的大小

按照占据空间从大到小的顺序渲染所有标签：

按照辐射状从内向外查找能放下该标签的坐标(x,y)：

先用粗粒度方法检测(x,y)是否能放下该标签，若不能检测下一个坐标(x',y')

否则用细粒度的抽样算法检测(x,y)是否能放下该标签，若能放下在(x,y)出渲染标签，否则检测下一个坐标(x',y')

EndWhile

计算完成之后按照 Base64 的编码格式将标签云传输到浏览器端解析显示。

4.1.8 览器照片显示模块

由于 flickr.com 传输数据有时候会延迟很长时间（比如 30 秒以上），而我们又没有照片的内容数据，所以不能通过服务器端实时去抓取需要显示的照片的内容数据，否则有可能会产生严重的延迟，所以将整个照片内容数据抓取流程放在浏览器端，在音乐计算完并开始播放后在后台通过 AJAX 请求需要展示的照片的内容数据。

4.2 音乐推荐流程

在介绍了 Picutne 的整体结构之后，本小结阐述整个音乐推荐的流程。如图图（4.2）所示，基于用户当前地理信息的 Pictune 音乐推荐系统具体包含如下步骤：

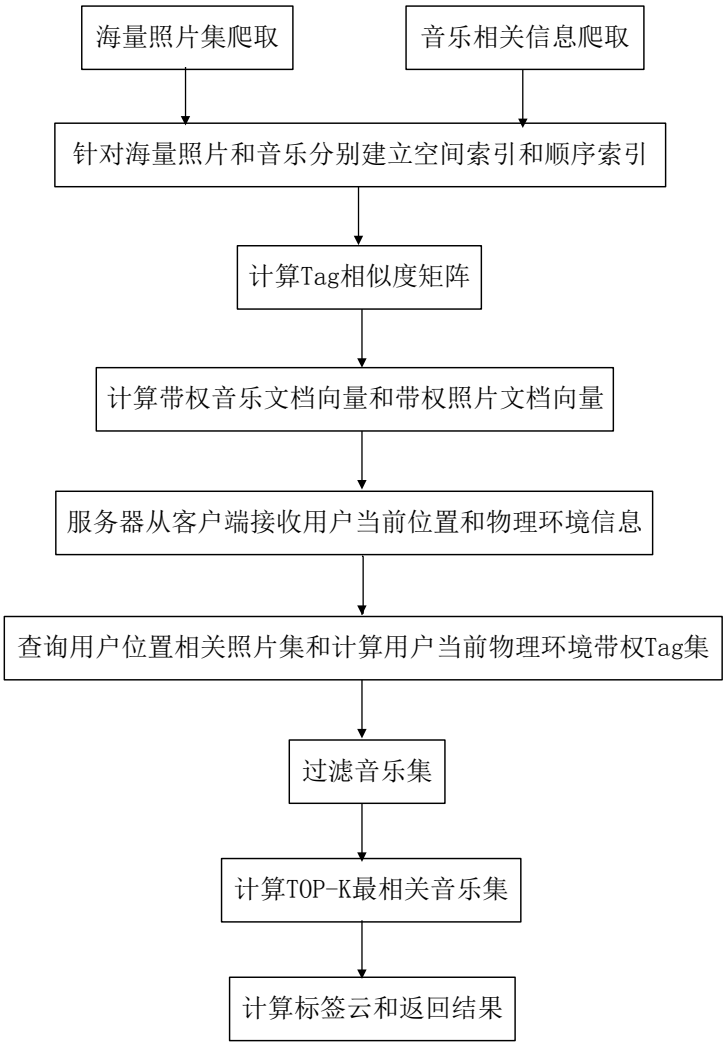

图 4.2 Pictune 系统推荐流程

步骤 1：从照片网站爬取具有 GPS 信息的照片集。

步骤 2：从音乐网站爬取包含歌词等相关信息的音乐集。

步骤 3：对步骤 1 得到的具有 GPS 信息照片集建立空间索引，在此过程中对每张照片的原始标签集进行分词处理得到规范的照片标签集。

由于照片数据量过于庞大，需要先对所有的照片进行预处理，提取出照片对

应的 GPS 坐标以及照片 ID、评论数、URL、标签集等等。然后根据 GPS 坐标插入到维护所有照片的空间索引中，比如以 R-TREE 为例（可以采用其他空间索引），R-TREE 当中维护了每张照片的 GPS 坐标以及照片的 ID，而照片的其余信息都存放在磁盘上。不论用户的位置信息表达为矩形还是圆形都可以通过层序遍历 R-TREE 节点做矩形或圆形求交运算获得该区域内的所有照片的数据。

以图（4.3）为例来进行说明。为方便说明，空间索引采用 R-TREE，图中的各个矩形就是 R-TREE 层次中的 MBR（最小包围矩形），内部节点包含下一层中的各个 MBR，依次递归包含。叶子节点中存储了位于叶节点 MBR 内的所有照片的 GPS 地理坐标和相应的照片 ID，图中每个小点代表一张照片。图中的虚线矩形表示用户当前的位置矩形，在查询的时候该位置矩形内所包含的照片组成了代表用户位置信息的照片集（图中用黑色小点表示这些照片），然后我们通过该照片集到磁盘上取得这些照片的标签集合以及相应的其他诸如评论、上传时间等等信息。

某些返回的照片标签有可能是由若干真正的标签拼接而成，采用 AC 自动机(Aho-Corasick string matching algorithm)将这些拼接而成的标签分词，得到照片对应的规范的标签集。

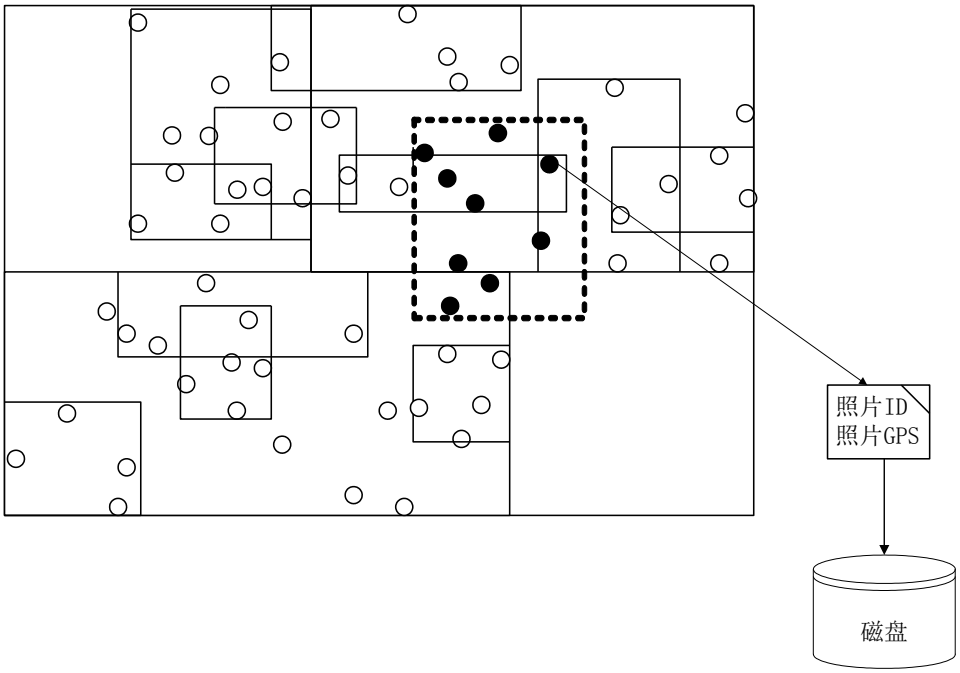

图 4.3 照片索引的一个例子

步骤 4: 对步骤 2 得到的包含歌词等相关信息音乐集建立存储索引，在此过程中对每首歌曲的歌词进行分词得到对应的音乐标签集和相应的音乐文档向量。

音乐信息按照音乐 ID 顺序存放在磁盘里，内存中开辟一个缓冲区存放一定比例的连续音乐信息，音乐信息的访问是按序的，每当需要访问下一部分的音乐信息时清空缓冲区，重新从磁盘读入下一部分的音乐信息。

通过英文分词技术将每首音乐的歌词进行分词得到对应的音乐标签集。

把所有歌词集里出现的每个不同的标签看作一个维度，每首音乐看作一个文档，把音乐表示成这个多维空间中的一个向量，每个维度上的值就是对应标签在该首音乐中出现的次数。

步骤 5：分析文档集得到一个标签相似度矩阵。

先分析文档集比如维基百科、大英词典中出现的所有英文文章，用一个滑动窗口得到标签对之间的共生次数，然后用 LIN 氏理论信息值为每个标签计算一个分布相似度向量，然后通过多次迭代最后产生一个类似 LSA 矩阵的相似度矩阵，其中每一行表示一个标签和其他标签的相似度向量，每一项就是两个标签之间的相似度，并按递减顺序排列。这样可以快速取得和每个标签最相似的标签。

为了方便每个标签最相似标签的查询，预先计算好每个标签最相似的若干标签并按照从高到低的顺序排列，生成一个相似矩形，存放于磁盘中，在程序运行期间加载到内存中。

步骤 6：通过查询步骤 5 得到的标签相似度矩阵将步骤 3 得到每张照片的标签集转换成位于音乐文档向量空间的照片文档向量。

在音乐匹配过程中需要计算音乐和一个照片标签集合之间的相似度，但是本质上照片标签集和音乐标签集（歌词分词后得到）是处于两个不同空间上的数据，要计算形似度就需要将它们转化到同一个空间中进行计算。方法是对于照片的每一个标签 x 计算出音乐标签集中和它最相似的标签 y，并用 y 代替 x。

比如如果要将照片中出现的标签 x 转换到一个音乐标签集中出现过的标签的话，首先将歌曲中出现的所有标签放到一个哈希表中，然后在标签相似度矩阵中 x 对应的数据行中从前往后扫描与 x 最相似的标签查看是否出现在音乐标签集的哈希表中，若找到则退出，否则 x 没有对应的音乐标签，也就不用考虑了。

处理完成之后，按照和步骤 4 类似的方法将每张照片的标签集转换成一个处于音乐向量空间的照片文档向量。

步骤 7：根据转换得到的照片集计算出照片生成模型中的参数（详见第七章“推荐算法”）。

步骤 8：客户端接收用户的当前位置信息和心情信息，传递给服务器进行计算。

用户通过客户端指定她所处的当前位置（可以是矩形或者圆形，长宽和半径都可更改，矩形和圆形的坐标都用经纬度表示），并选择自己的心情（通过文字选择或者图形化选择）和代表心情强烈程度的权重值（1 到 100），客户端同时计算出其他诸如天气状况，时间等等物理环境信息，客户端将这些信息传递给服务器。

步骤 9：服务器根据从步骤 8 得到的用户位置信息通过步骤 3 建立的照片集索引查询出用户位置包含的所有照片得到位置相关照片集，同时对从步骤 8 得到的物理环境信息进行加强得到代表用户当前物理环境的带权标签集。

按照递归查询的方法从照片空间索引中查询出用户位置包含的所有照片信息集。

用户的物理环境包括天气状况，时间，心情等等，其中用户心情表达为一个标签和对应的权重（1 到 100），权重越高表示用户当前的物理环境越强烈，并通过 WordNet-Affect 对用户当前的物理环境进行扩展：和用户当前物理环境类似的情绪的标签都被加入到表达用户的物理环境带权标签集里。

步骤 10：使用步骤 9 得到的代表用户物理环境的带权标签集对音乐进行过滤得到候选音乐集。

对于代表用户物理环境的带权标签集中的每一个标签，根据它对应的权值计算出一个等比例的 0-1 之间的阈值，使用步骤 5 中得到的标签相似度矩阵可以求得任何一首音乐中和它最相似的标签以及对应的相似度。对于任何一首音乐，如果这个带权标签集中任何一个标签在该首音乐中最相似的标签相似度没有超过对应的阈值就将这首音乐过滤掉，在音乐匹配过程中就不再考虑。

步骤 11：按照第七章中描述的推荐算法计算出和其最匹配的 K 首音乐；

若用户请求的是前 K 首最匹配的音乐，维护一个容量为 K 的最大堆，在每首音乐的匹配分数计算完毕之后相应地更新最大堆的内容，最后最大堆里记录的就是匹配度最高的 K 首音乐。

步骤 12：基于步骤 9 得到的位置相关照片集所包含的照片标签集计算出相应的标签云，最后将查询出的 K 首音乐和生成的标签云返回给客户端。

4.3 本章小结

本章首先介绍了 Pictune 音乐推荐系统的主要结构，然后详细阐述了音乐推荐的主要流程。

第5章 单词相似度计算

在上一章中详细解说明了 Pictune 音乐推荐系统的整个结构和推荐流程，本章将深入研究具体的单词相似度计算方法。本章首先说明单词相似度计算效率和效果对本文的重要意义，在此基础上给出几个相似度计算的方法，然后阐述 Pictune 系统里单词相似度计算的算法和单词转换算法。

5.1 单词相似度计算的重要性

为了完成音乐的推荐，我们需要通过移动用户主题和音乐文档相似度（Similarity）计算函数$\mathrm{sim}(u_{topic}, m_d)$计算出每首音乐和移动用户主题的相似度。而移动用户主题是由移动用户上下文生成，而后者是一个由 Flickr 照片集合采用 bag-of-words 模型生成的文本文档，该照片集合是根据用户的位置采用空间 KNN 查询或者空间范围查询从 Flickr 照片索引中实时查询得出的，所以最终这个集合生成的移动上下文包含的单词都只会是 Flickr 照片中出现过的单词。另一方面，音乐文档集主要是由音乐的歌词生成的，所以音乐文档集中出现的单词只会来自音乐的歌词中。

这样就出现了问题，我们在用相似度计算函数计算移动用户主题和音乐文档之间的相似度时就会产生很难匹配的问题。这主要是因为歌曲的歌词的用词习惯和 Flickr 用户对照片打标签的用词习惯差异很大。歌词往往是一句句连贯的句子，他们是通过句子的结构和语义性来表达歌曲的感情的，而 Flickr 照片的标签只是一个个没有结构的单词，用户是通过单词来表达情感的。

也可以说是音乐文档和移动用户主题处在两个差异很大的空间里面。如果硬要计算两者的相似度的话，效果会非常差，因为不论是采用哪种相似度度量函数，不管是 Cosine Similarity 或者 Pearson correlation 等度量函数都要依赖于相同单词域的值来做出计算和决策，如果两个空间的用词重叠度几乎为 0 的话，这些度量函数都很难做出正确有效的计算。

数据的稀疏性使得这个问题变得更加严重了。考虑 Flickr 网站的照片，虽然 Flickr 网站有着海量的照片（截止到本文完成时间为止，大约 70 亿左右），但是每张照片的标签数量很少，在本文中所考虑的照片集合中平均的标签数量为 13.2，这也说明了本文引入的移动用户主题的必要性：如果只是通过采样的方法得到若干照片，通过他们的元数据信息来得到移动用户上下文的话，单词数量过少，根本不能提供任何有用的信息。另外虽然每首音乐的歌词和标签合在一起组成的单词数量要远多于每张照片中的标签数量，但歌词中的句子大多都较短，而且经常重复，所以每首音乐对应的单词数量是很少的。

比如如果采用 TF-IDF 权值函数和向量空间模型，然后使用 Cosine Similarity 来计算两个空间中向量的相似度的话，因为所用单词差异较大，而且数据稀疏性会导致计算出的相似度都趋近于 0，这对于音乐推荐几乎是没有作用的。

综上，为了使得相似度计算是有效的，我们需要解决两个空间用词差异性问题。本文采用的方法是将其中某个空间的单词转换到另一个空间中的单词，这样就可以在同一个空间中处理问题了。但这产生了三个问题，一是如何转换的问题，二是转换的方向问题，应该从哪个空间向另一个空间转换；三是转换的效率问题。

首先考虑问题一，假设我们已经有了一个智能的能计算任何两个单词w_1，w_2相似度的函数$\mathrm{WordSim}(w_1, w_2)$，本文的算法是：假设音乐中出现过的所有单词的集合是M_w，Flickr 照片中出现过的所有单词的集合是P_w，然后假设所有音乐文档集合是 M，同时 Flickr 中的照片集合是 P，对于 P 中的每张照片$p_i = (id, l_d, \mathbf{w}_d)$（请参见第三章中的模型定义），我们将$p_i$的$\mathbf{w}_d$中的每个单词$w_{d,x}$都替换成$w_{d,x'}$，其中$w_{d,x'}$满足对于任何的单词$w \in M_w$，均有$\mathrm{WordSim}(w_{d,x}, w_{d,x'}) \geq \mathrm{WordSim}(w_{d,x}, w)$并且$w_{d,x'} \in M_w$。也就是说将每个单词$w_{d,x}$替换成音乐中出现过的和$w_{d,x}$相似度最高的单词。

这个过程可以离线预计算，将所有的照片中的单词替换成相似度最高的属于音乐空间的单词，这是因为在后期在线的推荐算法的计算中只需要访问已经转换过的单词而不需要再次访问转换前的单词，所以离线一次处理之后即可。另外第六章中所描述的推荐算法在建立 Flickr 照片的空间索引的时候采用的是增量式的

算法，从爬虫模块获得的照片均是通过增量式的方式插入空间索引中的，这也使得对照片单词的转换处理变得可以离线化：对于每张新的爬来的照片，首先通过WordSim相似度函数将其转换成音乐空间中的向量，然后通过增量式的插入算法插入到空间索引中。

这个算法的效率低下，因为对于每个待转换的单词我们都需要访问所有的音乐空间中出现过的单词，并且用WordSim函数计算两者之间的相似度。但是音乐单词的数量非常多，即使WordSim函数的计算开销很小上述代码的复杂度也太高。在介绍完本文中所采用的单词相似度计算函数之后会详细阐述另一个高效的实现版本。

对于问题二，考虑两个空间中包含的单词的总数，Flickr 照片中包含的不同单词的总数要小于音乐中出现的不同单词的数量，因此相对于从音乐空间转换到照片空间，从照片空间转换到音乐空间时成功的可能性更高：假设 Flickr 照片中具有的不同单词数是P_{num}，音乐空间中出现的不同单词数是M_{num}，如果$P_{num} < M_{num}$，并且假设和每个单词固有的、有效的相似单词平均数量是N，那么如果采用从音乐空间向照片空间的转换方向的话，每个单词转换的成功概率正比于$\frac{P_{num}}{N}$，反之采用向相反的方向的话，转换成功的概率正比于$\frac{M_{num}}{N}$，在$P_{num} < M_{num}$的情况下显然后者的概率更大，更容易转换成功。直观来说，将单词从照片空间向音乐空间的转化过程中，每个单词拥有的可选择性更多，转换得到与其最相似的单词的概率当然更大。

对于问题三，由于我们需要将所有涉及到的照片中的单词进行转换，整个过程需要转换的单词的数量非常多，而WordSim函数是其中的一个最基本的函数，在每次计算单词相似度的时候都会被调用，整个计算过程均依赖于它。因此即使转换流程可以离线计算，但是如果WordSim的调用需要消耗大量的时间的话整个过程仍然是无法忍受的；另外每次从爬虫获得了新的照片集之后都需要对索引进行更新，而更新之前都要进行单词转换，因此我们依然需要在线地进行单词转换，如果相似度计算过慢仍然会降低整个系统的处理效率。

下面将详细介绍本文设计的单词相似度计算方法。

5.2 词汇搭配（Collocation）

下面首先介绍词汇搭配的概念，然后讨论如何获得词汇搭配，最后说明如何使用词汇搭配设计一种简单的单词间相似度计算方法。

5.2.1 词汇搭配

从数据上来说要获得任意单词间的相似度的话，我们需要一个数据集，这个数据集必须包括所有英文词汇表中的单词，而且包含了这些单词的各种用法以及各种单词间的搭配方式，这样我们才能从中抽取出每个单词的特征，进而根据这些特征来计算出不同单词之间的相似度。考虑以上两个问题之后，本文最终采用的数据集是维基百科（Wikipedia）中所有的英文文章集，该文章集合几乎囊括了所有英文单词，而且包括各种丰富的词汇应用，具体的数据请参见第七章“实验结果”。

词汇搭配是一种常用的单词之间的组合方式，比如“weather a storm”（度过难关）、“file a lawsuit”（提起诉讼）和“the falling dollar”（美元汇率下跌）。虽然上述的词汇搭配很常用，但大多数的词汇搭配都非常特殊，通过语法分析或者语义分析都不能识别。比如虽然“baggage”和“luggage”是同义词，但是只有“baggage”可以被“emotional”（感情包袱）、“historical”（历史包袱）和“psychological”（心理包袱）修饰。

有研究指出[27]一个单词的意思在很大程度上是由该单词的词汇搭配的模式所决定的。虽然词汇搭配的知识非常重要，但是一般来说这些词汇搭配方式在手动编辑的词典中较少，因此本文选择从英文维基百科的文章中抽取这些知识。我们的目的在于从中抽取出涵盖面广而且精度高的词汇搭配。虽然词汇搭配一般来说具有重现的特性，但是一个特定的词汇搭配在一个常规大小的语料库中出现的次数并不一定较多。举个例子，在一个由华尔街日报（Wall Street Journal）和圣荷西水星报（San Jose Mercury）组成的有着 2200 万单词的语料库中，“emotional baggage”和“historical baggage”以及“psychological baggage”这三种词汇搭配都只出现了一次。所以为了取得较广的涵盖范围，即使一种搭配只出现了一次，也必

须提取出来。

5.2.2 词汇搭配的提取

词汇搭配表示为一个单词依赖三元组：它包括一个头部，一个依赖关系和一个修饰词。比如，从句子“I have a brown dog”中提取的三元组有：

（have V:subj:N I）
（have V:comp1:N dog）
（dog N:jnab:A brown）
（dog N:det:D a）

在上面的三元组中，依赖关系的标识符表示的意思请参见表（5.1）：

表 5.1 三元组中的依赖关系

依赖关系标识符	表达的依赖关系
N:det:D	一个名词(noun)和它的限定词(determiner)
N:jnab:A	一个名词(noun)和它的定语(adjective modifier)
N:nn:N	一个名词(noun)和它的名义修饰符(nominal modifier)
V:comp1:N	一个动词(verb)和它的宾语(noun object)
V:subj:N	一个动词(verb)和它的主语(subject)
V:jvab:A	一个动词(verb)和它状语(adverbial modifier)

我们的语料库是英文维基百科的所有文章，我们采用了解析器从中提取单词依赖三元组。我们采用了两个步骤来降低在解析过程中的出错数量。首先只有不超过 25 个单词长度的句子才会被交给解析器进行处理；其次只有解析完全的句子才会被包含进最后的输出中。

经过前面的步骤解析出了大量的单词依赖三元组，现在我们讨论用互信息（Mutual Information）来从单词依赖三元组中提取单词搭配。

首先一个单词依赖三元组(w_1, rel, w_2)可以被视为以下三个事件的共生事件：

A：随机选择一个单词w_1；

B：随机选择一种依赖关系rel；

C：随机选择一个单词w_2。

互信息比较两类信息，一是观测到的三元组的共生次数，二是一个基于独立性假设模型预测出的共生次数。在文章[28]中将三元组的互信息定义为：

$$\log \frac{P(A,B,C)}{P(A)*P(B)*P(C)}$$

从这个定义中可以知道，它隐式假设如果一个单词三元组并不表达一个真正的单词搭配的话，三个事件 A，B，C 则是独立的。但是这个隐式假设并不成立，因为三元组中两个单词的词性是由三元组中依赖关系类型所决定的。比如，如果 rel是 N:det:D 的话，则w_1必须是名词，而w_2必须是一个限定词。

因此本文做出另一个更为合理的独立性假设：在给定 B 的情况下，A 和 C 满足条件独立（Conditionally Independent）。如果采用图模型（Graphical Models）[29]来表示的话文章[30]中的假设和本文中采取的假设可以用贝叶斯网络（Bayesian Networks）方式表示如下：

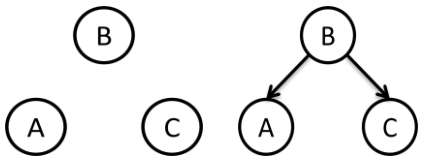

图 5.1 三元组对应的图模型

在本文条件独立的假设下，三元组(w_1, rel, w_2)的互信息可以重新定义如下：

$$\log \frac{P(A,B,C)}{P(B)*P(A|B)*P(C|B)}$$

上述定义中涉及的概率可以用三元组出现的频率来近似估计。但是一个重要的问题是：观测到的稀有事件的概率会被过高地估计，相反没有被观测到的稀有事件的概率会被低估。因此我们采用一个常量 c 对观测到的三元组(w_1, rel, w_2)的频率进行调整：

$$P(A,B,C) = \frac{\|w_1, rel, w_2\| - c}{\|*,*,*\|}$$

$$P(B) = \frac{\|*, rel, *\|}{\|*,*,*\|}$$

$$P(A|B) = \frac{\|w_1, rel, *\|}{\|*, rel, *\|}$$

$$P(C|B) = \frac{\|*, rel, w_2\|}{\|*, rel, *\|}$$

其中$\|w_1, rel, w_2\|$表示三元组(w_1, rel, w_2)在整个解析过后的语料库中出现的总次数。其中*表示在该维度对所有可能的单词或者关系类型进行频率求和。举个例子，$\|*, rel, *\|$代表所有三元组中单词间关系类型是rel的三元组出现的次数。

5.3 基于单词依赖三元组的单词相似度计算

根据第三节获得的单词依赖三元组已经可以计算任意两个单词之间的相似度了。在本节中我们详细讨论该计算方法。

对于每一个单词依赖三元组(w_1, rel, w_2)，可以把它看作单词w_1和w_2的某个特征（feature）维度。比如，对于（avert, V:comp1:N, duty）这个三元组，我们说“duty”这个单词具有特征 obj-of(avert) ，这是因“duty”是“avert”的宾语，同样的“avert”具有特征 obj(duty)。其他的具有特征 obj-of(avert)的单词包括“default”，“crisis”，“eye”，“panic”，“strike”，“war”等等。

从解析得出的所有三元组中我们为每个单词提取对应的特征。表（5.2）给出了单词“duty”和“sanction”的部分特征。每一行对应一维特征。“duty”和“sanction”所在列里的标号“X”表示相应单词具有“X”所在行的特征。在解析得到的语料库中，所有的可作为“include”的名词主语的单词都具有特征“subj-of(include)”。同理，所有可以被“fiduciary”修饰的名词都具有特征“adj(fiduciary)”。

表 5.2 单词特征包含信息量的例子

特征(Feature)	duty	sanction	$-\log P(f)$
f1：subj-of(include)	X	X	3.15
f2：obj-of(assume)	X		5.43
f3：obj-of(avert)	X	X	5.88
f4：obj-of(ease)		X	4.99
f5：obj-of(impose)	X	X	4.97
f6：adj(fiduciary)	X		7.76

f7：adj(punitive)	X	X	7.10
f8：adj(economic)		X	3.70

单词之间的相似度可以根据它们的特征来计算。本文中的相似度度量函数采用的是文章[31]中提出的度量方法。该方法定义两个物体之间的相似度为两个物体包含的公共的信息量和两个物体包含的总信息量的比值。如果用 F(w)来表示物体 w 包含的特征集的话，两个单词的相似度定义如下：

$$\mathrm{sim}(w_1, w_2) = \frac{2 * I(F(w_1) \cap F(w_2))}{I(F(w_1) + F(w_2))}$$

上式中 I(S)表示特征集 S 中包含的信息量。如果我们假设特征之间都是独立的话，有：

$$I(S) = -\sum_{f \in S} \log P(f)$$

特征 f 的概率 P(f)可以如此估计：设所有包含特征 f 的单词总数是f_{num}，这些单词的词性是 s，如果词汇集 V 中包含的所有词性为 s 的单词的个数是s_{num}，那么 P(f)就等于$\frac{f_{num}}{s_{num}}$。举个例子，解析后的语料库中一共有 32868 个不同的名词，其中有 1405 个名词被用作过“include”的主语，所以特征 subj-of(include)的概率是$\frac{1405}{32868}$；但是特征 adj(fiduciary)的概率却只有 14/36868，这是因为只有 14 个不同的名词被“fiduciary”修饰过。我们可以得出结论特征 adj(fiduciary)比特征 subj-of(include)包含的信息量要大得多。这其实和我们的直觉是一样的，因为如果我们说一个词可以被 fiduciary 修饰，这句话包含的关于这个单词的信息量要远远高过这个词可以作为 include 的主语这句话所包含的信息量。

表（5.2）中的标注为$-\log P(f)$的列中就显示了每个特征所包含的信息量。如果表（5.2）中的特征是单词“duty”和“sanction”包含的所有特征的话，这两个单词之间的相似度可以计算如下：

$$\mathrm{sim}(\text{"duty"}, \text{"sanction"}) = \frac{2 * I(\{f_1, f_3, f_5, f_7\})}{I(\{f_1, f_2, f_3, f_5, f_6, f_7\}) + I(\{f_1, f_3, f_4, f_5, f_7, f_8\})}$$

这样计算出来的和“duty”最相似的前 60 个单词是：

responsibility 0.13, position 0.10, sanction 0.10, tariff 0.09, obligation 0.09, fee 0.09, post 0.08, job 0.08, role 0.08, tax 0.08, penalty 0.08, condition 0.07, function 0.07, assignment 0.07, power 0.07, expense 0.07, task 0.07, deadline 0.07, training 0.07, work 0.07, standard 0.06, ban 0.06, restriction 0.06, authority 0.06, commitment 0.06, award 0.06, liability 0.06, requirement 0.06, staff 0.06, membership 0.06, limit 0.06, pledge 0.06, right 0.05, chore 0.05, mission 0.05, care 0.05, title 0.05, capability 0.05, patrol 0.05, fine 0.05, faith 0.05, seat 0.05, levy 0.05, violation 0.05, load 0.05, salary 0.05, attitude 0.05, bonus 0.05, schedule 0.05, instruction 0.05, rank 0.05, purpose 0.05, personnel 0.04, worth 0.04, jurisdiction 0.04, presidency 0.04, exercise 0.04

单词“duty”在 WordNet 里有三种不同的意思：(a)责任(responsibility)，(b)工作、责任(work,task)，(c)关税表，收费表(tariff)，这些单词都已经包含在了上面的单词集中。

5.4 优化方法

前文的方法具有如下明显的缺点：

上面的方法中第一步需要从语料库中抽取所有的单词搭配，而这需要对语料库中的所有句子进行词法、语法分析，但解析器为了得到一个全局的解析，往往会做出很多很差的局部解释，比如对于语义上本身就比较有歧义的单词选择了错误的词性。如果这个语料库本身是一个人工编撰的词典的话，这个问题会变得尤其严重，因为词典里会包括各种晦涩难懂的词汇用法。

举个例子来说明这个问题。如果使用 WordNet 作为语料库的话“job”和“class”可以作为动词，而“cancel”可以作为名词。假设有一个句子包括“hold jobs”，因为“hold”和“jobs”都可以被当作名词和动词，所以解析器必须考虑如下两种情况：

（1） 动词“hold”把“jobs”当作它的宾语。

（2） 名词“hold”修饰另一个名词“jobs”。

（3） 名词“hold”是动词“jobs”的主语。

解析器会选择这三种关系里面哪一种决定于在当前句子里谁更匹配剩下的部分。总之，单词相似度最终结果的好坏很大程度上决定于解析的准确度，但至少在目前来看，解析器效果仍然不够理想。

另外语法解析的速度太慢，如果要经常切换语料库的话，计算代价太高。

本节提出的优化方法将解决前面提到的问题。

5.4.1 单词特征提取

在前面提到的方法中单词的特征提取依赖的是解析器对单词依赖三元组的提取，本节将采用不依赖于解析器的方法提取类似三元组的结构。

本质上，三元组结构表达了单词之间的语法结构，但语法结构具有一定的特征，在一个语法结构内部单词之间的相对距离基本都是一定的，直观想法是利用单词之间的距离来代表单词所处的语法结构。

本节对单词特征的提取方法如下：

（1） 首先对语料库进行预处理，将所有文章进行分词得到单词（Tokenize），在此过程中忽略所有的停用词（Stop Word）。

（2） 用一个半径为 Win 的滑动窗口对所有处理过的文章顺序处理，在滑动的过程中计算单词对之间的共生次数（Co-Occurrence）。在计算共生次数的时候要将单词在相应滑动窗口中的相对位置记录下来，这样做是为了模拟单词依赖三元组的结构，也可以看作是对这些单词的语法依赖关系的粗糙近似。比如对于某个句子，若它包含了单词三元组（donut，obj-of，eat），滑动窗口方法将得到类似的三元组（donut，-2，eat），表示在滑动窗口里“eat”出现在“donut”前面两个单词的位置。所以现在单词的特征已经不再和 bag-of-words 模型中一样是一个个的单词，而是一个单词和相对位置距离组成的有序对。这样的上下文表达方式比简单的滑动窗口方法更能表达单词之间的上下文信息，也就能更好地用于单词相似度计算中。

（3） 通过前面两步的处理，我们得到一个共生矩阵（Co-Occurrence Matrix）。

共生矩阵每一行描述了一个单词，称作原始零阶单词向量（Raw Zero Order Word Vector）。但是共生矩阵的纬度不再是 v*f（v 是语料库中除开停用词之后不同单词的个数，f 是每个单词使用的特征数，请参见第七章），而是 v*f*r（r 是滑动窗口的大小），共生矩阵中的每一项储存了单词对之间在滑动窗口中某个距离下出现的总次数。比如代表“donut”的一行中包括（donut，-2，eat）这一项，对应的值为 43，表示在处理完整个语料库后，在滑动窗口中“eat”出现在“donut”前面 2 个单词位置处的这种情况一共碰到了 43 次。

（4） 将（3）中得到的共生矩阵进行处理。对其中的每一项采用前面提到的三元组互信息公式进行转换得：

$$\mathrm{MI}(w,r,w') = \frac{P(w,r,w')}{P(r)*P(w|r)*P(w'|r)}$$

$$= \frac{\frac{\|w,r,w'\|-c}{\|*,*,*\|}}{\frac{\|*,r,*\|}{\|*,*,*\|}*\frac{\|w,r,*\|}{\|*,r,*\|}*\frac{\|*,r,w'\|}{\|*,r,*\|}}$$

$$= \frac{(\|w,r,w'\|-c)*\|*,r,*\|}{\|w,r,*\|*\|*,r,w'\|}$$

其中 w 和w'均代表相应的单词，r 代表滑动窗口中对应的相对距离（也就是单词依赖关系的近似表达），$\|w,r,w'\|$代表这种关系出现的次数，c 是前文提到的为了解决稀有事件的概率会被过高估计，没有被观测到的稀有事件的概率会被低估这一问题引入的调整常数。

经过上面的步骤之后我们最终获得了一个处理过的共生矩阵，每一行代表一个单词，包含了用互信息处理之后和其他单词之间的信息量，称作零阶单词向量（Zero Order Word Vector），它包含了描述一个单词的所有特征，在下文中用它来计算单词之间的相似度。

5.4.2 单词相似度计算

经过前文的处理后，每个单词都用一个零阶单词向量描述，直观的，两个单

词之间的相似度体现在代表它们的两个向量之间的相似度，所以我们用文章[32]中提出的衡量两个特征向量间相似程度的 Lin 氏度量方法来计算两个单词之间的相似度如下：

$$\mathrm{Lin}(w_1, w_2) = \frac{\sum_{(r,w)\in T(w_1)\cap T(w_2)} (f(w_1, r, w) + f(w, r, w_2))}{\sum_{(r,w)\in T(w_1)} f(w_1, r, w) + \sum_{(r,w)\in T(w_2)} f(w, r, w_2)}$$

$$\text{其中 } T(w) = \{(r, w') \ s.t \ f(w, r, w') != 0\}$$

公式 5.1

上市中$f(w_1, r, w_2)$表示w_1的零阶单词向量中代表 (r, w_2)的位置处的值。$T(w_1)$代表w_1的零阶单词向量中所有$f(w_1, r, w_2)$值不为 0 的(r, w_2)所组成的集合。该 Lin 氏相似度和 Cosine Similarity 比较类似，但零阶单词向量并没有归一化，而且采用的是加法而不是乘法；另外它和 Jaccard 相似度（Jaccard Similarity）也比较相似，但 Jaccard 相似度中每一项是无权的。

举个例子，用 Lin 氏距离算出的和单词“palm”最相似的前 12 个单词是：

palms (0.1345) coconut (0.1059) olive (0.0870) pine (0.0823) citrus (0.0745) oak (0.0677) mango (0.0652) cocoa (0.0645) banana(0.0627) bananas (0.0623) trees (0.0570) fingers (0.0560)

这样对于每一个单词都可以算出和它最相似的若干个单词，我们取相似度不小于一个阈值 Th 的所有单词和相应相似度组成一个新的向量，这个向量称作一阶单词向量（First Order Word Vector）。一阶单词向量和零阶单词向量不同，它不仅包含了和某个单词出现在同一滑动窗口里的单词，还包括能出现在相似上下文中的单词。

根据这些一阶单词度向量我们再一次用 Lin 氏距离计算单词之间的相似度，得到单词间的二阶相似度（Second Order Similarity），而前面根据零阶单词向量计算得到的相似度是一阶相似度（First Order Similarity）。二阶相似度的计算不仅考虑了单词之间的共生关系，还考虑了单词之间语义的可替换性，因为它考察了单词出现语境的相似性。另外可以看出 5.4 节中的方法可以看作是本节中的一阶相似度。

5.5 优化方法实现

在 5.4 节中给出了单词相似度计算的优化方法，这一节在此基础上详细讨论这一优化方法的各种实现，并分析各个版本的优缺点。由于最终我们需要二阶的单词相似度，所以这里我们只讨论二阶单词相似度计算的实现，实际上一阶单词相似度计算的方法和二阶几乎是一样的。

在讨论不同的实现方法之前先说明一下输入输出中设计的问题。

关于输入。在计算两个单词之间的二阶相似度的时候，主要的两个操作是获取单词对应的一阶单词向量和求两个一阶单词向量之间的交集。在所有单词的一阶单词向量计算完成之后，每个单词的一阶单词向量平均大概包含了 6000 左右的单词量，那么所有一阶单词向量总共就包括了大概 $1.2*10^9$ 的单词量，数据过于庞大。为了提高程序的效率，这些庞大的一阶单词向量的储存是一个比较大问题，下面的实现中会针对该问题做出不同的取舍。

关于输出。在我们的推荐系统中，单词相似度的计算是一个最基本的操作，涉及到的计算次数非常庞大，所以需要单词相似度的计算复杂度能达到 O(1)。所以本节中的实现对每个单词 w 来说，将计算出和它相似度最大的 Top-K 个单词，并储存下所有这些单词和相应的二阶相似度，这样在计算单词之间的相似度时，就能直接取用这些已经预计算好的数据，而不用再次计算，将计算复杂度降到了最低。上面提到的 Top-K 的 K 值大小和 w 对应的一阶单词向量包含的单词个数相等，这是因为我们做出一个了假设：对于单词 w，如果其对应的一阶单词向量所包含的单词组成的集合是 FW，那么我们在计算二阶相似度时只需要考虑 FW 中的单词，原因是 FW 中的单词是所有和 w 一阶相似度超过某个阈值（比如 0.01）的所有单词组成的集合，这里的假设也就是说二阶相似度高的单词它们的一阶相似度也不会太低。

所以输出也和输入有着相同的问题，那就是我们需要输出的二阶单词对的个数也达到了 $1.2*10^9$ 的数量级。但是和输入不一样，这里我们决定按照矩阵的形式存放所有计算出的二阶单词向量，该矩阵每行对应一个单词的二阶单词向量，但它是一个斜矩阵，每一行的长度不一定相等，而且每一行是按照二阶单词向量

中的二阶单词相似度按照降序排列的，如图（5.2）所示。

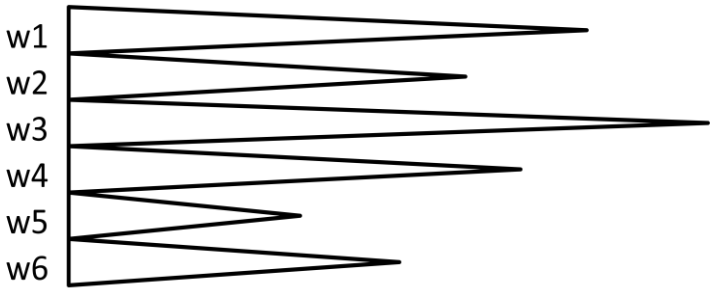

图 5.2 二阶单词相似度矩阵

下面各小结对复杂度的讨论中假设单词的个数为 N，每个单词对应的一阶单词向量的平均长度为 L。

5.5.1 基于 Apache Lucene 全文检索引擎的实现版本

这里把所有输入的一阶单词向量存放于 Lucene 索引中。索引中的每个 Document 对应一个单词，而每个 Document 内部包括两个域（Filed），其中一个域对应一阶单词向量中所有的单词，而另一个域对应一阶单词向量中单词的一阶相似度。

在计算两个单词w_1 和w_2之间的二阶相似度时，首先从索引中查询出两个单词对应的 Document，然后从 Document 中取出一阶单词向量和对应一阶相似度，然后根据公式（5.1）计算两者的相似度。完整计算所有二阶单词向量的伪代码如下所示：

```
Algorithm1: LuceneBased(LI,W)
Input: Lucene index LI, Word set W
Output: Second-order word similarity matrix SWM
for every word w in W:
    sv={} #second-order vector of w initialized to empty vetor
    fw_w = read first-order word vector from LI of w
    fs_w = read first-order similarity vector from LI of w
    #use fw_w as key and fw_v as value to build a hash set
    hw = hash set(fw_w, fw_v)
    for every word v in fw_w:
        fw_v = read first-order word vector from LI of v
        fs_v = read first-order similarity vector from LI of v
        n = sum of every first-order similarity value of word in both fw_v and hw
        d = sum of(fs_w) + sum of(fs_v)
        sv.add(v, n/d)
    endfor
    #sort second-order vetor into nonincreasing order
    sort(sv)
    #set row i of SWM to sv
    SWM(i:) = sv
endfor
```

该实现中计算复杂度为 $O(NL^2)$，占用内存 O(L)，对 Lucene 索引的查询访问次数为 $O(NL^2)$，IO 次数 $O(NL^2)$。

5.5.2 常驻内存的实现版本

Lucene 版本的实现中，由于索引是存放于磁盘上的，所以每次都需要首先通过索引找到某个单词对应的 Document，然后从磁盘中取出相应的一阶单词向量，磁盘访问和 Lucene 查询是非常慢的，而且我们需要计算大概 $1.2*10^9$ 个单词对之间的相似度，即需要至少 $1.2*10^9$ 数量级的磁盘访问和 Lucene 查询，因此虽然 Lucene 版本的实现消耗的内存非常小，但是 IO 却成为了瓶颈。另外，因为 Lucene 是文本索引引擎，所以每次查询时还要把字符串转换为浮点数，也引入了额外的代价。

最后通过哈希表计算两个一阶单词向量之间相交部分的和也不够高效：每次都需要先对其中一个向量建立哈希表，然后扫描另一个向量检测是否存在于哈希表中。

在第 7 章中的实验中可知 Lucene 版本的实现中一秒钟大概只能计算出 80 对左右的二阶单词相似度。因此 Lucene 版本实现只有当内存不够，而且只需要计算少量单词间二阶相似度的时候有使用价值。

为了减少磁盘 IO 和 Lucene 查询代价，本小结里，先把 Lucene 索引中的数据提取出来重新组织成一个矩阵存放在磁盘上，在计算二阶相似度时先将该矩阵读入内存，然后处理。为了快速计算两个单词一阶向量的交集，该矩阵的每一行都是按照单词的 ID 升序排列，这样在计算两个单词一阶向量交集的时候只需要使用两个指针从两个向量开始往后扫描，哪个指针指向 ID 较小就将此指针往后移动，相等时即找到了一个共同单词。

将 Lucene 索引转换成矩阵的操作所需计算量为 O(NLlogL)，内存占用 O(L)，IO 次数 O(NL)，Lucene 查询 O(N)。

完整计算所有二阶单词向量的伪代码如下所示：

```
Algorithm2: MemoryResident(LI,W)
Input: First-order word similarity matrix FWM, Word set W
Output: Second-order word similarity matrix SWM
for every word w in W:
    sv={} #second-order vector of w initialized to empty vetor
    fw_w = read first-order word vector from FWM of w
    fs_w = read first-order similarity vector from FWM of w
    #use fw_w as key and fw_v as value to build a hash set
    for every word v in fw_w:
        fw_v = read first-order word vector from FWM of v
        fs_v = read first-order similarity vector from FWM of v
        i = j = n = 0
        while i<len(fw_w) and j<len(fw_v):
            if fw_w.wordId < fw_v.wordId: ++i
            elseif fw_w.wordId > fw_v.wordId: ++j
            else n+=fw_w.val, ++i ++j
        endwhile
        d = sum of(fs_w) + sum of(fs_v)
        sv.add(v, n/d)
    endfor
    #sort second-order vetor into nonincreasing order
    sort(sv)
    #set row i of SWM to sv
    SWM(i:) = sv
endfor
```

该实现中计算复杂度为 $O(NL^2)$，占用内存 $O(NL^2)$，IO 次数 O(NL)。

5.5.3 基于多核并行计算的实现版本

常驻内存的实现版本中，会把所有单词的一阶单词向量全部读入内存中，内存消耗太大，但是计算速度大大提高（请参见第 7 章，整个二阶相似度矩阵计算时间约为 63 个小时）。

本小结节将结合前面两种方法的优点，提出一种速度快而且内存占用较小的方法。

在二阶相似度计算的过程中，有很多的并行性可以利用，比如在计算第 1 个单词的二阶单词向量时可以同时计算第 2 个单词的二阶单词向量，如果核够多的话还可以同时计算第 3 个单词的二阶单词向量，等等。另外每个二阶单词向量的输出并不和其他的二阶单词向量的输出有冲突，这更加说明了并行计算的可能性。假设一共有 C 个核可用，理论上该方法可以将计算时间缩短到原有时间的$\frac{1}{C}$。

为了减小程序占用的内存大小，不能一次性将一阶单词矩阵读入内存，我们的方法是在内存中维护一个大小为 T（代表 T 个单词对应的一阶单词向量）的哈希表，哈希表中的数据来自每个线程从磁盘中读入内存的一阶单词向量。如图（5.3）所示，对于每个核来说，每次计算当前单词和下一个单词 v 的二阶相似度的时候，v 的一阶单词向量有大约$\frac{T}{N}$的概率已经在哈希表中了，如果不在哈希表中的话就需要从磁盘上读取，然后添加到哈希表中。另外对哈希表中的所有数据用一个 LRU 队列维护，如果在从磁盘读取单词 v 对应的一阶向量到内存里的时候哈希表已经没有空闲位置的话，就将当前没被访问中的最久前被访问的一阶单词向量替换，只要 T>C，则必存在这样一个一阶单词向量。

如图（5.3）所示，白色矩形代表每个单词对应的一阶单词向量，它们的长度可以不同，所有黑色矩形代表处在哈希表中的一阶单词向量对应的单词，当然每个核对应的处在哈希表中的单词的个数也可以不同，而且这些单词也并不一定连续。

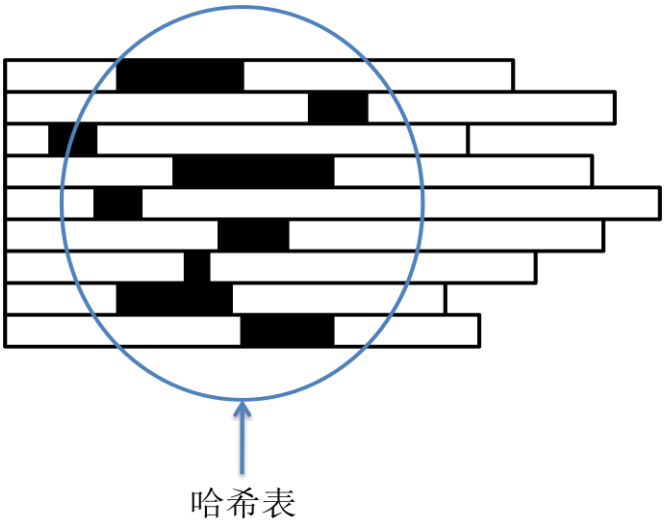

图 5.3 多核并行计算方法图示

该实现中计算复杂度为$O(NL^2)$，占用内存$O(TL)$，IO 次数大约为$O(NL(1-\frac{T}{N}))$。

为了达到最好的性能，磁盘操作应该越小越好，也就是 T 越大越好，T 的大小受着内存大小的制约，如果假设每个数据段的大小是 F，内存大小为 MEM，那么需要满足$TLF \leq MEM$，那么最优的 T 值大概为$\frac{MEM}{LF}$。

从第七章中可以看出本实现非常高效，将整个计算时间缩短了十几倍。

5.6 照片单词集转换

有了 5.5 节中处理得到的二阶单词相似度矩阵之后，在 5.1 节中提到的对照片单词进行转换的过程就可以做得更加高效了。在对一个单词w进行转换的时候，我们从二阶单词相似度矩阵里就能知道与其相似度最高的单词集S_w（不同w对应的S_w的大小有可能不同），而且S_w中的单词是按照和单词w相似度的大小由高到低排好序的。假设音乐元数据中包含的所有单词集合是M_w,那么我们可以对S_w从头开始扫描直到找到第一个出现在M_w中的单词w'为止，那么w'必然是M_w中和w最相似的单词。由于S_w（一般 3000-4000）的大小比M_w（二十几万）小得多，该方法效率明显优于 5.1 节中的方法。

对每张照片的单词转换处理函数的伪代码如下所示：

```
Algorithm3: TransformPhotos(P,SWM,M)
Input: Photo set P, Music word set M, Second-order sord similarity matrix M
Output: Transformed photo set P'
#cache stores all the transformed words for later use
cache = make emtpy hashset
for every photo p in P:
    for every word w in p:
        if w is in cache:
            replace w with cache(w)
        else:
            SV = get second-order vector of w in SWM
            find first word w' in SV which occurs in M
            replace w with w'
            cache(w) = w'
        endif
    endfor
    store transforme photo in P'
endfor
```

如果我们将二阶相似度矩阵放在磁盘上的话，整个处理流程的时间复杂度为$O(\min(|P_w|, W) * |\overline{S_w}| + |P_w|)$，其中$|P_w|$是整个照片集包含的单词总数，W是$|P_w|$中包含的不同单词的数量，$|\overline{S_w}|$是$S_w$的平均大小值；占用内存$O(\max(|S_w|) + M_w)$；IO 次数$O(\min(|P_w|, W))$。

5.7 本章小结

本章首先说明单词相似度计算效率和效果对本文的重要意义，在此基础上给出几个相似度计算的方法，最后给出了基于二阶相似度矩阵的照片集转换方法。

第6章 推荐算法

在前面章节的基础上，本章详细阐述本文中使用的推荐算法。首先 6.1 节中给出基于地理主题模型的推荐算法概述，然后介绍照片生成模型，包括模型参数的估计、复杂度分析，最后给出基于该模型的音乐推荐算法。

6.1 基于地理主题模型的推荐算法概述

本节先给出算法的主要想法和机器学习模型，然后再给出该模型中的参数估计方法。然后说明如何计算最优的 Top-K 首音乐。

主要想法是根据地理区域的划分，不同的区域描述着不同的主题，在本文中这些主题体现在每张照片的元数据中，我们假设每张照片都描述了一个主题，照片间的主题有可能相同，也可能不同，在地理上距离近的照片更有可能是描述同一主题的。同样的，每首音乐也表达了一个主题。推荐的时候根据用户的当前位置和范围可以知道用户的移动用户地理主题，然后查询哪些音乐包含的主题和当前的移动用户地理主题最相关，就推荐该首音乐。

详细来说，是把每张照片看作一个由基于地理位置的文档生成模型生成出来的文档，这个模型里面有各种模型参数，这些参数中包括了和定理位置相关的参数，而且这些参数决定了每张照片的生成概率，由于我们已经有了大量的照片数据，由这些数据我们可以倒着推回去，计算出模型的参数估计；在推荐音乐的时候把每首音乐看作一个文档，用户当前位置作为模型的输入参数，对每首音乐文档计算该生成模型产生它的 likelihood，然后取产生 likelihood 最大的 Top-K 首音乐作为推荐结果。

为了更好描述下文中的算法，这里先给出地理主题分布的概念：

定义 6.1 地理主题分布（Geographical Topic Distribution）：形式化地，$p(z|l)$代表地点$l=(x,y)$产生主题 z的概率，其中x,y分别代表经纬度，而且$\sum_{z\in Z} p(z|l)=1$。从$p(z|l)$我们可以知道哪些主题在地点l比较热。

另外假设整个照片集合具有固有的K个主题，它们形成了一个主题集合$Z = \{\theta_z\}$，每个主题描述的信息是一个定义在单词集合上的单词分布，即对于任意主题z，它的表达方式为：$\theta_z = \{p(w|z)\}_{w\epsilon V}$ 满足$\sum_{w\epsilon V} p(w|z) = 1$。举个例子，比如某个主题 z_{food} 描述的是关于 food 的主题，那么它的组成可能就是 p(香蕉|z_{food})=0.4，p(苹果|z_{food})=0.3，p(火龙果|z_{food})=0.1，p(芒果|z_{food})=0.1，p(榴莲| z_{food})=0.1，很明显主题 z_{food} 描述的最多的信息是关于香蕉的。

根据生成模型，这固有的 K 个主题能生成整个单词集合和整个照片集合。比如对于单词“香蕉”，它可能包含于主题 $z_{植物}$和主题 $z_{食物}$。

为了发掘和计算出地理主题分布，我们需要一个模型来描述单词地理分布的空间结构。在空间上相隔较近的单词更有可能属于同一个主题。为了表达这个概念，我们假设存在若干个区域（Region），而地理主题分布是由这些区域而非文档产生的。也就是说如果两个单词在空间上相隔较近，则它们更有可能属于同一个区域；如果两个单词属于同一区域，它们更有可能在描述同一主题。

举个例子，比如钓鱼岛的主题就是{“战争”、“争端”…}，而浙江的主题就是{“好声音”、“西湖”…}。不过某个地理区域包含的主题不一定唯一，所以和上面的 topic 定义类似，区域 r 的表达方式为$\emptyset_r = \{p(z|r)\}_{z\epsilon Z}$ 满足$\sum_{z\epsilon Z} p(z|r) = 1$。比如钓鱼岛现阶段具有两个主题，一个是{“战争”、“争端”…}占 70%，另一个是{“钓鱼”、“打捞”…}占 30%。

和地理主题不同的是每个区域具有地理信息，所以产生两个问题，一是同一区域里产生同一主题的概率在不同位置应该不同（离区域热点近的位置具有主题 z 的概率应该更高），二是区域不一定具有规则的形状。问题一可以假设同一区域的分布服从二维高斯分布；但考虑问题二应该假设区域服从高斯混合模型（Gaussian Mixture Model）分布，这样才能模拟不规则形状的分布（实际上就是任意二维形概率状分布都能用一定数量的椭圆概率分布逼近）。

在下一小节中我们会详细讨论照片的生成模型。

6.2 照片生成模型

和 Latent Dirichlet Allocation 类似，我们假设所有照片都是由一个生成模型生成的。并且假设空间中一共包含了 N 个区域 K 个地理主题。所有 K 个地理主题的分布参数组成集合$\theta = \{\theta_z\}_{z\epsilon Z}$，所有 N 个区域组成的集合是 R。我们假设每个区域服从二维高斯分布，参数化为$(\mu, \Sigma) = \{(\mu_r, \Sigma_r)\}_{r\in R}$，其中$\mu_r$,是区域 r 高斯分布的均值向量，而$\Sigma_r$是区域 r 的协方差矩阵。另外$\alpha$是所有区域服从的离散分布参数，并且满足$\sum_{r\in R} p(r|\alpha) = 1$。另外每个主题是由区域生成的，所以假设所有区域里主题服从的分布参数为$\emptyset = \{\emptyset_r\}_{r\in R}$,其中每一个$\emptyset_r$服从分布$\emptyset_r = \{p(z|r)\}_{z\in Z}$，其中$p(z|r)$是区域r产生主题z的概率，对于每一个区域r有$\sum_{z\in Z} p(z|r) = 1$。

在我们的模型里面，每个主题都是由相应的区域产生的，而不是由文档产生的，而且每一个区域里数据的分布服从高斯混合模型。在空间上分布较近的单词更有可能属于同一区域。我们的生成模型产生一张照片$(id, l_d, \mathbf{w}_d)$的过程如下：

（1） 从区域服从的离散分布$r\sim Discrete(\alpha)$中抽样出一个区域r，α表示每一个区域的重要性，$Discrete(\alpha)$是根据区域的重要性而产生的分布。

（2） 从区域 r 服从的二维高斯分布$p(l_d|\mu_r, \Sigma_r)$中抽样出点l_d。其中有：

$$p(l_d|\mu_r, \Sigma_r) = \frac{1}{2\pi\sqrt{|\Sigma_r|}} \exp\left(\frac{-(l_d - \mu_r)^T \Sigma_r^{-1} (l_d - \mu_r)}{2}\right)$$

公式 6.1

（3） 对于$\mathbf{w}_d$中的每一个单词w：

（a） 首先从多项分布$\emptyset_r$分布中抽样主题z，概率是$p(z|r)$。

（b） 再从多项分布θ_z中抽样单词w，概率是$p(w|z)$。

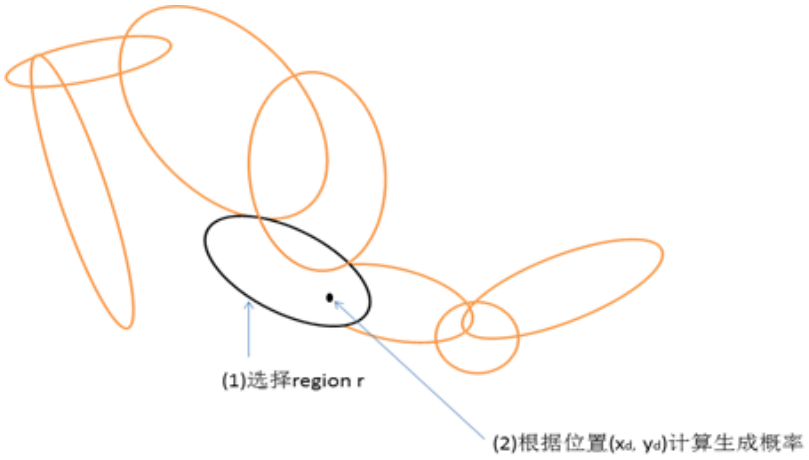

图 6.1 照片生成模型图示

在这个模型中每一个主题不仅仅和其中一个区域有关联，而是可以同时属于若干个区域，这一点和高斯混合模型类似，所以我们的模型可以处理分布不是规则形状的主题。在这个模型里对每个区域的识别既使用了地理信息也使用了文本信息，并同时挖掘出和区域相应的地理主题分布。

把所有的模型参数写为 $\Psi = \{\theta, \alpha, \emptyset, \mu, \Sigma\}$。给定所有的照片集合 $\{(id, l_d, \mathbf{w}_d)\}_{d\in D}$，则整个数据的对数似然函数（log-likelihood）如下：

$$\begin{aligned} L(\Psi; D) &= \ln p(D|\Psi) \\ &= \ln \prod_{d\in D} p(\mathbf{w}_d, l_d|\Psi) \\ &= \sum_{d\in D} \ln(\sum_{r\in R} (p(r|\alpha) * p(\mathbf{w}_d, l_d|r, \Psi)) \end{aligned}$$

公式 6.2

现在我们已经有了大量照片的数据集合$\{(id, l_d, \mathbf{w}_d)\}_{d\in D}$，根据我们的模型知道这些数据背后有很多的隐含变量，我们现在要极大化公式（6.2）中的对数似然函数。但是根据公式（6.2），ln 函数里包含了一个求和运算，所以如果我们通过求导使得$L(\Psi; D)$等于 0 来求解最大值的话我们得不到一个公式解。下一小节中我们使用期望最大化（Expectation Maximization）来求解最大值，并求出取得最大对数似然函数时各个变量的值，但 EM 算法只能求得局部最优解，所以求得的参数值是估计值而不一定是精确值。

6.2.1 模型参数估计

在本小结中我们使用 EM 算法迭代计算，每次迭代都使得对数似然函数的值得到提高。

如果假设r_d是照片$d = (id, l_d, \mathbf{w}_d)$所在的区域，但实际上我们并不知道该照片所属的区域，所以这是一个隐含变量。因此我们引入隐含变量$p(r|d, \Psi)$表示给定照片 d 和参数Ψ时，照片 d 所在区域r_d是 r 的概率。

在 E-step 里，我们首先更新隐含变量$p(R|D, \Psi)$的后验概率分布（Posterior Distribution），在 M-step 里，因为我们并不知道$p(R, D|\Psi)$的分布，为了最大化$\ln p(D|\Psi)$，我们使用 E-step 中得到的$p(R|D, \Psi)$后验概率分布来计算$\ln p(R, D|\Psi)$在该概率分布下的期望值$E_{p(R|D,\Psi)}\{\ln p(R, D|\Psi)\}$来代替$\ln p(D|\Psi)$，然后在每一次迭代的 M-step 里最大化该期望值。

6.2.1.1 E-step

根据 Bayes 公式我们可以计算出$p(r|d, \Psi)$的后验概率值为，其中参数右上角的 t 代表参数所在的迭代轮数：

$$p(r|d, \Psi^t) = \frac{p^t(r|\alpha)p(\mathbf{w}_d, l_d|r, \Psi^t)}{\sum_{r' \in R} p^t(r'|\alpha)p(\mathbf{w}_d, l_d|r', \Psi^t)}$$

其中$p(\mathbf{w}_d, l_d|r, \Psi^t)$计算方式如下：

$$p(\mathbf{w}_d, l_d|r, \Psi^t) = p(\mathbf{w}_d|r, \Psi^t)p(l_d|r, \Psi^t)$$

公式 6.3

其中$p(l_d|r, \Psi^t)$的分布如公式（6.1）所示。

$p(\mathbf{w}_d|r, \Psi^t)$服从多项分布，等于：

$$p(\mathbf{w}_d|r, \Psi^t) \propto \prod_{w \in \mathbf{w}_d} p(w|r, \Psi^t)^{c(w,d)}$$

公式 6.4

其中$c(w, d)$代表单词w在照片d中出现的次数。和文章[33]一样，对于每个单词w我们假设它由一个混合模型生成，包括一个背景模型（Background Model）和我们的区域模型。使用背景模型的目的是让主题逐渐趋向于使用更有区分度的单

词。根据这个混合模型有：

$$p(w|r,\Psi^t)=\lambda_B p(\omega|B)+(1-\lambda_B)\sum_{z\epsilon Z}p^t(w|z)p^t(z|r)$$

公式 6.5

其中$p^t(w|z)$由分布θ^t给出，而$p^t(z|r)$由分布$\emptyset^t$给出，而$p(\omega|B)$由背景模型给出，如下所示：

$$p(\omega|B)=\frac{\sum_{d\in D}c(w,d)}{\sum_{w\in V}\sum_{d\in D}c(w,d)}$$

6.2.1.2 M-step

在引入了隐含变量$p(R|D,\Psi^t)$之后，$\ln p(R,D|\Psi^t)$在$p(R|D,\Psi^t)$的分布概率下的期望值是：

$$\begin{aligned}E_{p(R|D,\Psi^t)}\{\ln p(R,D|\Psi)\}&=E_{p(R|D,\Psi^t)}\{\ln(\prod_{d\in D}p(r|\alpha)p(\mathbf{w}_d,l_d|r,\Psi))\}\\&=E_{p(R|D,\Psi^t)}\{\sum_{d\in D}\ln p(r|\alpha)p(\mathbf{w}_d,l_d|r,\Psi)\}\\&=\sum_{d\in D}\sum_{r\in R}p(r|d,\Psi^t)\ln p(r|\alpha)+\\&\quad\sum_{d\in D}\sum_{r\in R}p(r|d,\Psi^t)\ln p(l_d|r,\Psi)+\\&\quad\sum_{d\in D}\sum_{r\in R}p(r|d,\Psi^t)\ln p(\mathbf{w}_d|r,\Psi)+\end{aligned}$$

公式 6.6

从公式（6.6）知道，要最大化$E_{p(R|D,\Psi)}\{\ln p(R,D|\Psi^t)\}$可以分别最大化相加的三个部分，因为涉及到的参数$\Psi=\{\theta,\alpha,\emptyset,\mu,\Sigma\}$中，$\alpha$分布在第一个相加量里，$\mu,\Sigma$分布在第二个相加量里，而$\theta,\emptyset$分布在最后一个相加量里，这三个相加量可以说是独立的，所以分别最大化这三个量就相当于最大化了个函数。

（1） 首先我们最大化$\sum_{d\in D}\sum_{r\in R}p(r|d,\Psi^t)\ln p(r|\alpha)$：

由于$\sum_{r\in R}p(r|\alpha)=1$所以为了最大化上式，要使用拉格朗日乘子$\lambda$。所以变成了我们要最大化下式：

$$\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)\ln p(r|\alpha)+\lambda\left(\sum_{r\in R} p(r|\alpha)-1\right)$$

将上式对$p(r|\alpha)$求导得到：

$$p(r|\alpha)=\frac{\sum_{d\in D} p(r|d,\Psi^t)}{-\lambda}$$

公式 6.7

而对λ求导得到：$\sum_{r\in R} p(r|\alpha)-1$

将公式（6.7）带入上式得到：

$$\frac{\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)}{-\lambda}=1$$

得到：$-\lambda=|D|$，带入公式（6.7）得到：

$$p^{t+1}(r|\alpha)=\frac{\sum_{d\in D} p(r|d,\Psi^t)}{|D|}$$

公式 6.8

（2） 最大化$\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)p(l_d|r,\Psi)$：

$$\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)p(l_d|r,\Psi)$$

$$=\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)\frac{1}{2\pi\sqrt{|\Sigma_r|}}\exp\left(\frac{-(l_d-\mu_r)^T\Sigma_r^{-1}(l_d-\mu_r)}{2}\right)$$

对μ_r求导得到：

$$\mu_r^{t+1}=\frac{\sum_{d\in D} p(r|d,\Psi^t)l_d}{\sum_{d\in D} p(r|d,\Psi^t)}$$

公式 6.9

对Σ_r求导得到：

$$\Sigma_r^{t+1}=\frac{\sum_{d\in D} p(r|d,\Psi^t)(l_d-\mu_r{}^t)(l_d-\mu_r{}^t)^T}{\sum_{d\in D} p(r|d,\Psi^t)}$$

公式 6.10

（3） 最大化$\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)\ln p(\mathbf{w}_d|r,\Psi)$：

$$\sum_{d\in D}\sum_{r\in R} p(r|d,\Psi^t)\ln p(\mathbf{w}_d|r,\Psi)$$

$$=\sum_{d\in D}\sum_{r\in R}\sum_{w\in V} c(w,d)p(r|d,\Psi^t)\ln(\lambda_B p(\omega|B)+(1-\lambda_B)\sum_{z\in Z} p(w|z)p(z|r))$$

为了最大化上式，我们使用另一个 EM 算法来最大化它：

$$\varphi(w,r,z)\leftarrow\frac{(1-\lambda_B)p(w|z)p(z|r)}{\lambda_B p(\omega|B)+(1-\lambda_B)\sum_{r\in R} p(w|z)p(z|r)}$$

公式 6.11

$$p(z|r)\leftarrow\frac{\sum_{w\in V} c(w,d)p(r|d,\Psi^t)\varphi(w,r,z)}{\sum_{z'\in Z}\sum_{w\in V} c(w,d)p(r|d,\Psi^t)\varphi(w,r,z')}$$

公式 6.12

$$p(w|z)\leftarrow\frac{\sum_{d\in D} c(w,d)p(r|d,\Psi^t)\varphi(w,r,z)}{\sum_{w'\in V}\sum_{d\in D} c(w',d)p(r|d,\Psi^t)\varphi(w',r,z)}$$

公式 6.13

其中$\varphi(w,r,z)$是隐含变量，表示区域r里出现的单词w的主题是z。

6.2.2 模型参数估计复杂度

在 E-step 的时候需要$O(KN|V|)$复杂度去计算公式（6.5）中所有(w,r)的$p(w|r,\Psi^t)$值，其中K是主题的个数，N是区域的个数，$|V|$是单词集的大小。按照公式（6.4）对所有(d,r)计算其$p(\mathbf{w}_d|r,\Psi^t)$值需要$O(N|W|)$的复杂度，其中$|W|$是所有照片中包含的标签的数量。另外还需要$O(|D|)$复杂度对所有照片计算$p(l_d|r,\Psi^t)$。所以为了得到所有(r,d)的$p(r|d,\Psi^t)$值需要复杂度$O(KN|V|+N|W|)$。

在 M-step 中，从公式（6.8）、（6.9）、（6.10）知道需要$O(N|D|)$复杂度更新$p^{t+1}(r|\alpha),\mu_r^{t+1},\Sigma_r^{t+1}$的值。从公式（6.11）、（6.12）、（6.13）可知为了更新θ^{t+1}和$\emptyset^{t+1}$的值需要复杂度$O(T_2KN|V|)$，其中T_2是在计算公式（6.11），（6.12），（6.13）时的迭代次数。所以 M-step 的总复杂度是$O(N|D|+T_2KN|V|)$。

综上总的复杂度是$O(T_1(KN|V|+N|W|+N|D|+T_2KN|V|))$。其中$T_1$是总的迭代次数。

6.3 推荐算法

本小结先阐述推荐算法的细节，然后分析其复杂度。

6.3.1 算法描述

有了上面的模型，现在我们对整个空间进行划分，对得到的每个子区间利用上面的模型参数估计方法计算出其对应的模型参数。在推荐音乐的时候，首先找到移动用户所在区间，然后做出相应的推荐。具体描述如下：

首先根据照片的 GPS 坐标将照片全部插入 R^*-tree 或者四叉树中，一直划分到每个叶子节点里的照片数量小于等于一个系统常数 THRESHOLD 为止，并对得到的每个子区间利用上面的模型参数估计方法计算出其对应的模型参数。假如最后得到了 Q 个区间，那么我们最终会计算得出 Q 组参数$\{\Psi_1, \Psi_2 \ldots, \Psi_Q\}$，每组参数都是根据该区间内包含的照片元数据计算得到的。

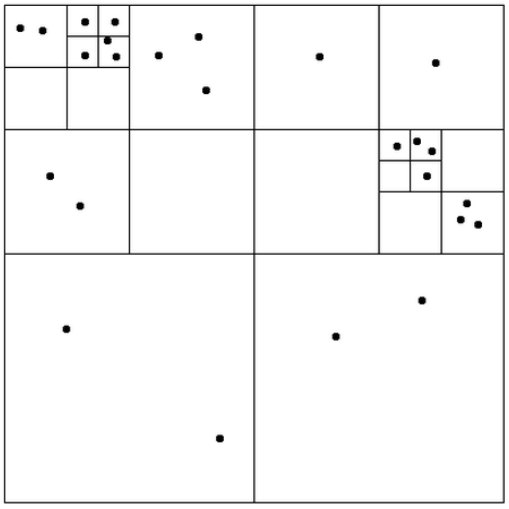

图 6.2 照片空间划分图示

假设用户在位置$l_{user} = (x_{user}, y_{user})$提交了请求，假如$l_{user}$处在区间 q 里面，那么我们能从预先计算出的参数里得到该区间对应的参数Ψ_q，现在如果我们把每首音乐均看作是带有用户当前地理位置的文档的话，那么在我们的生成模型的基础上就能对每首形如$(l_{user}, \mathbf{m}_d)$的音乐计算出一个 likelihood 值（其中$\mathbf{m}_d$是根据音乐歌词生成的第 d 首音乐的文档），likelihood 值排名最高的 Top-K 首音乐就是在位置l_{user}向用户推荐的 K 首音乐。原因如下：

因为Ψ_q是我们根据区间q里的照片的元数据信息根据我们的模型计算出来的参数值，如果我们把音乐也看作和照片具有同样形式的数据的话，likelihood 越高的音乐说明其和当前环境越搭配，也就是和用户当前位置所在区间的主题越相关。

按照第三章中的定义有：

$$sim(u_{topic}, l_{user}, \mathbf{m}_d) = p(\mathbf{m}_d, l_{user} | \Psi_q) = \sum_{r \in R} (p(r|\alpha) * p(\mathbf{m}_d, l_{user} | r, \Psi_q)$$

公式 6.14

其中Ψ_q隐含在参数u_{topic}中，$p(\mathbf{m}_d, l_{user} | r, \Psi_q)$可由公式（6.3）计算求得。

综上，在推荐的时候顺序扫描每一首音乐，计算它们的sim值，最后将具有最大的 K 个sim值对应的音乐推荐给用户。

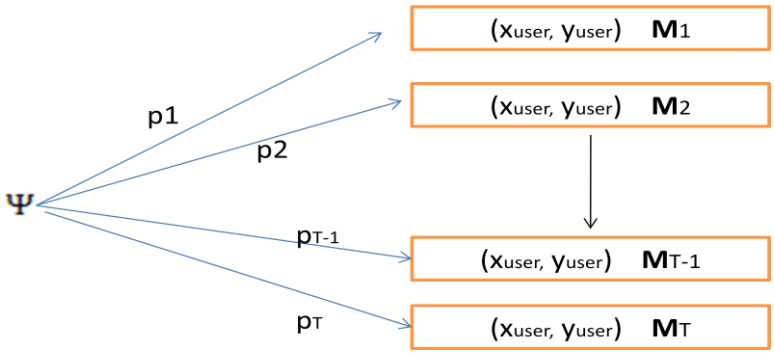

图 6.3 推荐算法图示

6.3.2 复杂度分析

由于我们对每首音乐都要计算一次sim值，所以最终的复杂度和计算sim的复杂度成正比。根据公式（6.1）、（6.3）、（6.4）可以得到：

$$p(\mathbf{m}_d, l_{user} | r, \Psi_q) = \prod_{w \in \mathbf{m}_d} (p(w|r, \Psi_q)^{c(w, \mathbf{m}_d)}) \, p(l_{user} | r, \Psi_q)$$

公式 6.15

由于我们处在二维空间里面，所以$p(l_{user} | r, \Psi_q)$可以看作常数时间，计算$p(w|r, \Psi_q)$的代价为K，K为主题的个数，所以对$\mathbf{m}_d$计算 sim值的复杂度为$O(KN|\mathbf{m}_d|)$，$|\mathbf{m}_d|$为音乐文档$\mathbf{m}_d$中包含的单词数量。所以总的复杂度为$O(KN|\mathbf{M_d}|)$，其中$|\mathbf{M_d}|$表示所有音乐文档包含的单词数量。

在一个典型的应用场景中K大概等于 10，N等于 30，音乐数量为 5 万，假设每首单词数在 100 左右，那么计算量大概为$1.5 * 10^9$，完全可以满足要求。另外如果我们对每首音乐都预处理一次，只保留不同单词以及它出现的次数的话，复杂度还能继续降低，因为我们可以用$\log c(w, \mathbf{m}_d)$的时间计算出$p(w|r, \Psi_q)^{c(w,\mathbf{m}_d)}$，而不是耗费$c(w, \mathbf{m}_d)$的时间。

最后，可以发现公式 6.15 中只有其中$p(l_{user}|r, \Psi_q)$这一项和用户当前的地理位置有关，而因子$\prod_{w \in \mathbf{m}_d}(p(w|r, \Psi_q)^{c(w,\mathbf{m}_d)})$完全可以在对区间 q 预处理时对每首音乐预先计算存储，在实时推荐的时候再取出来，这样公式（6.15）的计算复杂度就降低到了$O(1)$，$sim(u_{topic}, l_{user}, \mathbf{m}_d)$的计算复杂度降低到了$O(N)$，进而使得总的推荐复杂度降到了$O(|M|N)$，$|M|$表示总的音乐数量，所以计算量大概为$1.5 * 10^6$，降低了 3 个数量级，所以这是一个非常高效的算法，而且不论用户在什么地方计算复杂度都是稳定的，引入的额外空间代价也很小，我们只需在每一个区间里面对每一个区域 r 额外存储一个值，所以额外空间代价为$O(QN|M|)$，完全可以忍受。

6.4 本章小结

本章首先提出了照片生成模型的概念，然后使用了 EM 算法对模型参数进行了估计，最终在此模型的基础上设计了一个高效的推荐算法，并分析了它的复杂度。

第7章 实验结果

本文首先在第四章介绍了整个系统的结构，然后在第五章介绍了单词相似度计算的两种算法，并讨论了优化算法的三种实现方法，分别是基于 Lucene 的方法、常驻内存的方法、多核并行计算的方法。随后在第六章描述了基于地理主题的推荐算法。本章通过实验对这些算法的性能和效果进行评估。

7.1 实验环境和设置

本实验中，数据爬虫（包括照片元数据、音乐元数据、音乐内容数据）使用 Python 实现，单词相似度矩阵使用 Java 和 C++语言实现，整个推荐系统使用 Java 实现。实验运行在一台具有 4 个物理 Intel® Xeon® E7420 2.13GHZ 的 CPU、16 核、64GB 内存的 Linux 服务器上。另外地理主题模型的参数预计算使用 Matlab 实现，计算平台为 Intel® Core™2 Duo CPU E7500 2.93GHZ，4.00GB 内存。

7.2 原型 Demo 展示

如图（7.1）所示是我们做出的推荐系统原型展示，模拟移动用户所在的环境和服务器端进行交互。整个系统分为后台服务器和前端页面展示，架构为 B/S 结构，开发 IDE 为 Myelipse 8.5。后台服务器开发语言为 Java 和 Python，容器为 Apache Tomcat 7.0.26，运行环境为 Linux，前端页面展示使用 Javascript 和 Html5。

页面左端是控制面板，用户可以选择当时的心情状态，一共包含了“OPEN”、“HAPPY”、“ALIVE”、“GOOD”、“LOVE”、“INTERESTED”、“POSITIVE”、“STRONG”、“ANGRY”、“DEPRESSED”、“CONFUSED”、“HELPLESS”、“INDIFFERENT”、“AFRAID”、“HURT”、“SAD”这 16 种心情，它们均是来自于 Wordnet-Affect[5]语料库中的单词，另外用户可以选择自己此时心情的强弱程度，可选范围为[0,100]，数值越大表明该心情越强烈。

页面中间的红框表示用户选择的模拟自己所在位置的区域，点击页面下端的

[5] wndomains.fbk.eu /wnaffect.html

“Start”按钮之后服务器就会计算出相应的 Top-K 首音乐，返回给页面展示在页面的右端，并按照相关度从高到低排列；用户所在位置会显示在页面的右下位置处；页面下端的播放器自动从服务器端获取数据并自动播放排名最高的音乐，另外可以点击页面右方的音乐列表切换播放的音乐；在页面中间会展示一张该区域内在 Flickr 网站上点击率最高的照片，并且展示一张代表这个区域内所有照片元数据信息的标签云，标签云中的单词的字体大小和该单词在该区域内出现的权重有关；在页面的左方还会显示另外三张点击率最高的照片以及以往推荐过的历史信息。

本推荐原型平均响应速度为 1 秒钟以内。

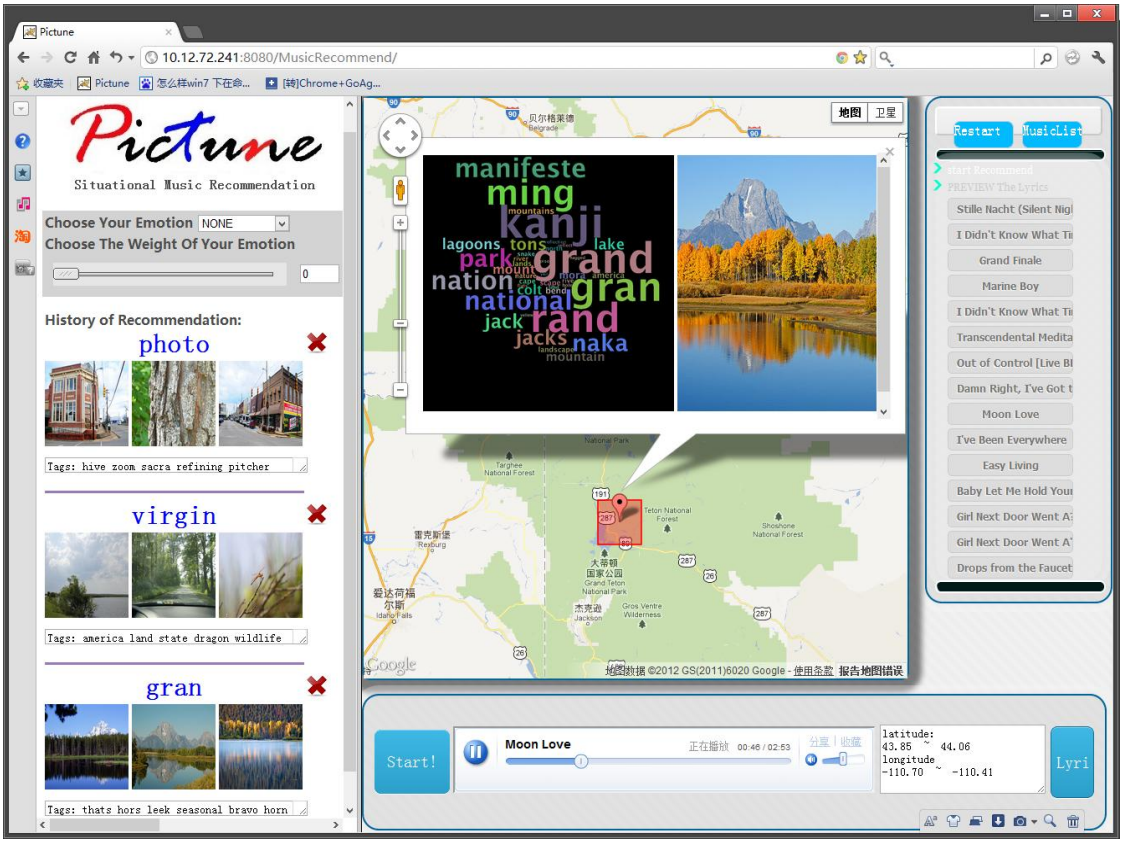

图 7.1 Pictune 原型图示

7.3 实验数据获取

本文中涉及到的数据包括照片网站中的照片元数据、音乐网站中的音乐内容数据、音乐社区的音乐元数据以及英文维基百科文档数据。

下面分别描述这些数据的获取方法。

7.3.1 照片元数据的获取

为了能根据照片元数据信息获得移动用户上下文和移动用户地理主题，我们需要大量的带有 GPS 坐标的照片。幸运的是，带有这些信息的在线照片的数量近年来逐渐增多。本文中的照片元数据来自于 Flickr 照片分享网站[6]。Flickr 网站是世界上最大和最好的线上照片分享和管理的网站。现在大概有 70 亿张照片，其中有上亿的照片带有 GPS 坐标信息。另外每张照片的元数据信息包括 ID、标题、标签、GPS 坐标、评论、描述、拍摄时间、上传时间、喜欢数、查看数等等。本文中的照片元数据均由爬虫程序结合 Flickr 网站提供的 API[7]获取。

最后获取的所有照片的相关数据如下表所示：

表 7.1 照片集统计数据

照片数量	71123221
平均标签数量	13.2
不同标签数量	93204
爬取消耗时间	93 小时

这 7000 多万张照片如果一张张按照 ID 爬取的话，因为每张照片需要大概 2 秒钟时间下载，那么总共要消耗大概 4 万个小时。本文使用的方法是使用多线程和进行批量下载：

当前的系统只推荐英文音乐，首先准备一个全世界著名地区的名字表，比如对于美国来说就包含了 50 州的名字，对于中国就包含了 34 个省的名字，每个名字表示我们想要爬取的照片所在的区域。Flickr 网站提供了一个 API 函数叫做 flickr.photos.search，它的参数中包括所搜索照片中包含的标签这一项，每次抓取的时候将前面准备的地区名字表传入，并指明所需照片必须包含 GPS 坐标（因为我们需要知道每张照片拍摄的具体位置）。Flickr 数据库就会一次性返回照片元数据信息中包含了此标签的前 4000 张符合要求的照片。但是这个函数有一个漏洞，

[6] www.flickr.com
[7] http://www.flickr.com/services/api/

如果一次返回的照片数量多于 1500 左右，那么后面返回的照片其实和前面的照片有重复了，为了克服这个漏洞，在传入的参数中设定照片上传时间的下限值和上限值，并采用二分搜索的方法来计算这个间隙的最优值使得每次返回的照片数量接近 1000 左右。并采用多线程加速的方法，最终抓取照片的速度可以达到大概 500 张每秒，一共消耗了接近 100 个小时的时间爬取。

当然，这个方法会抓取很多重复的照片，比如对于列表中的名字“USA”和“Portland”，有些照片可能会同时包含这两个标签。所以最后再进行一遍数据清洗只保留 ID 不同的照片。最后形成一个拥有 7000 多万张照片的照片元数据数据库。

图（7.2）是所有照片在地图上的分布示意图，图中浅蓝色的点表示该位置拍摄的照片，我们从所有照片中抽样了十分之一，大概 700 万张照片用以制作图（7.2）：

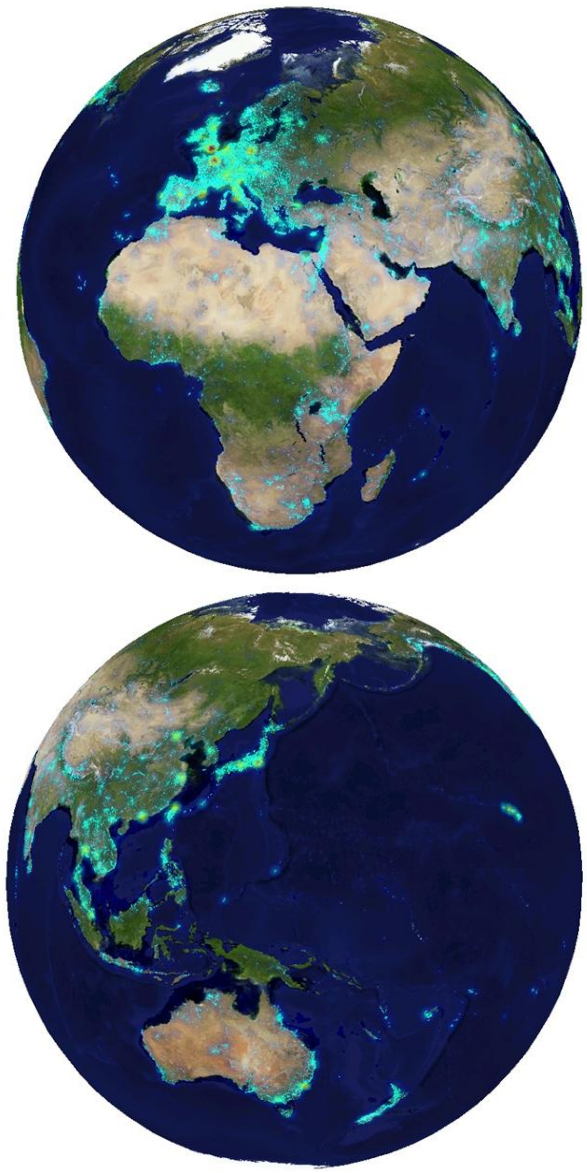

图 7.2 照片数据分布示意图

7.3.2 音乐元数据的获取

因为我们对音乐文档的生成是根据音乐的元数据（包括音乐名字、歌手名和标签以及歌词）进行计算获得的，所以我们需要建立一个包含大量英文音乐的数据库。

音乐名字、歌手名和歌词是从 lyrics 网站[8]通过爬虫爬取的。但是和照片爬取

[8] www.lyrics.com

流程不同，该网站没有类似 Flickr 网站的 API 系统，经过研究发现只能一首首地抓取元数据，方法是申请一个 User Key，然后通过如下的连接访问描述每首音乐的页面：

http://www.lyrics.com//user//api//1418012//open_track//T%20+音乐 id

这样会从服务器得到一个 html 文件，然后使用正则语言可以得到这首音乐对应的歌手名、歌曲名以及歌词。并且使用多线程加速，最后收集了一个具有十几万首歌的歌词的数据库。另外为了保证音乐的质量，我们对收集到的所有音乐进行了一遍清洗，所有歌词中包含单词数量小于 20 的音乐都被我们删掉了，并且对于音乐名和歌手名均相同的音乐只保留一份（我们假设它们是同一首音乐）。经过这些处理之后最后剩下了大概 6 万首有效的音乐。

音乐的标签从网站 Allmusic[9]获取。该网站提供了 API 供接入。对于上面爬取的所有有效音乐，我们通过 Allmusic 网站的 API 获取对应音乐的标签，这些标签包括音乐的心情标签、主题标签。

7.3.3 音乐内容数据的获取

上节所获得的所有音乐组成了我们系统的音乐元数据数据库，每首音乐都有可能会被我们的系统所推荐，所以为了能播放相应的歌曲，我们需要所有歌曲的内容数据。由于我们一共有大概 6 万首音乐的元数据，这个数据量比较庞大，如果手工下载这些音乐的话几乎是不可能完成的任务，另外音乐的版权也是一个重要的问题。最终我们决定从百度 MP3[10]通过程序多线程地下载程序。

可是，百度 MP3 并没有提供公开的 API 系统，无法直接从它的数据库中下载。经过研究百度 MP3 播放音乐的百度音乐盒发现，百度 MP3 有个不公开的访问音乐文件的方式：假设我们需要的音乐名字是 TITLE，歌手名字是 SINGER，那么通过访问地址 http://box.zhangmen.baidu.com/x?op=12&count=1&title=TITLE$$SINGER$$$$可以间接地获得音乐内容数据。

举个例子，假设我们需要下载歌手“齐秦”的“大约在冬季”这首音乐，那么通

[9] www.allmusic.com
[10] http://music.baidu.com/?from=new_mp3

过请求地址 http://box.zhangmen.baidu.com/x?op=12&count=1&title=大约在冬季$$齐秦$$$$，我们可以从服务器得到一个 XML：

```
1.  <?xml version="1.0" encoding="gb2312" ?>
2.  <result>
3.      <count>1</count>
4.  <data>
5.  <encode>http://song.feifa-radio.com/Q/20050701/jingxuan/YjI$.Wma</encode>
6.  <decode>1.Wma</decode>
7.  <type>2</type>
8.  <lrcid>49684</lrcid>
9.      </data>
10. </result>
```

其中的 count 值为 1 是说返回的是一首歌曲信息，encode 的值是歌曲加密后的地址,加密只是对文件名加密的,我们需要的只是前面的路径,也就是 http://song.feifa-radio.com/Q/20050701/jingxuan/ 这部分,然后复制 decode 的值: 1.Wma 与前面的相拼就是正确的下载地址:

http://song.feifa-radio.com/Q/20050701/jingxuan/1.Wma

type 的值为 2 表示此歌曲文件类型是 wma 的,其它的 1 表示 rm,0 表示 MP3, 通常我们下载的类型都是 MP3 或 WMA 的,所以只需要有 2 或 0 的。因为在歌曲抓取模块中已经维护了一个歌曲名、歌手名数据库，所以对这个数据库进行遍历，每首歌都采用上面的方式得到一个 XML，然后分析 XML 得到每首歌的下载地址，最后用程序下载。用多线程加速后每秒钟的下载速度能达到 2-3MB 每秒，去除掉百度 MP3 里没有的歌曲，最终建立一个拥有 5 万左右歌曲的数据库。

7.3.4 英文维基百科文档数据的获取

本文在计算单词相似度时使用的语料库是英文维基百科中的文章。语料库从 http://dumps.wikimedia.org/enwiki/获得，包含了大概 30 万左右的英文文章和大约 2.67 亿左右的单词。对于其中单词出现次数小于 100 次的单词我们不予以考虑，最后一共有 202578 个有效的单词，另外在介绍单词相似度计算的优化方法时我们提到过共生矩阵里面的每一行的维度是 f*r，而不是 f，其中的 f 是使用的作为特征的单词的数目，我们将出现次数最多的前 100000 个单词作为特征单词（即 f=100000），出现次数排名 100000 的单词出现的总次数是 254 次。

7.4 二阶单词相似度矩阵计算三种方法性能测试

本组实验将详细考察和对别基于 Lucene 索引的方法、常驻内存的方法以及多核并行计算的方法在计算二阶单词相似度矩阵时的性能。对性能的比较主要从计算时间和占用内存两方面考虑。

7.4.1 不同方法计算时间比较

如图（7.3）所示，比较的是三种方法各自计算 3200 个单词对应的二阶单词向量所消耗的时间，也就是大约 1.92 千万个单词对之间二阶相似度所消耗的时间，单位是分钟。其中 MultiCore 方法使用的是无内存限制的版本，也就是一次性将一阶单词相似度矩阵读入内存的版本。

从图中可以看出，LuceneBased 方法消耗的时间远远大于其余两个方法，这是因为每次计算单词对之间的相似度时都要对 Lucene 索引进行两次访问和查询，IO 代价太高，然后通过哈希表计算相似度，也引入了额外的计算开销；MemoryResident 方法通过预计算将 Lucene 索引中的数据顺序存放在一个矩阵当中，然后将整个一阶相似度矩阵导入内存大大减小了 IO 的代价，而且因为每个单词对应的一阶单词向量已经按照单词的 ID 从小到大排列，也大大加快了一阶单词向量对之间的集合求交计算，所以总的处理时间极大降低，将计算时间从 3900 分钟降低到了 45 分钟；最后无内存限制版本的 MultiCore 方法通过将计算平分到 16 个核并行计算，再一次大大降低了处理时间，最终将 3200 个二阶单词向量的计算时间优化到了 2.9 分钟左右。

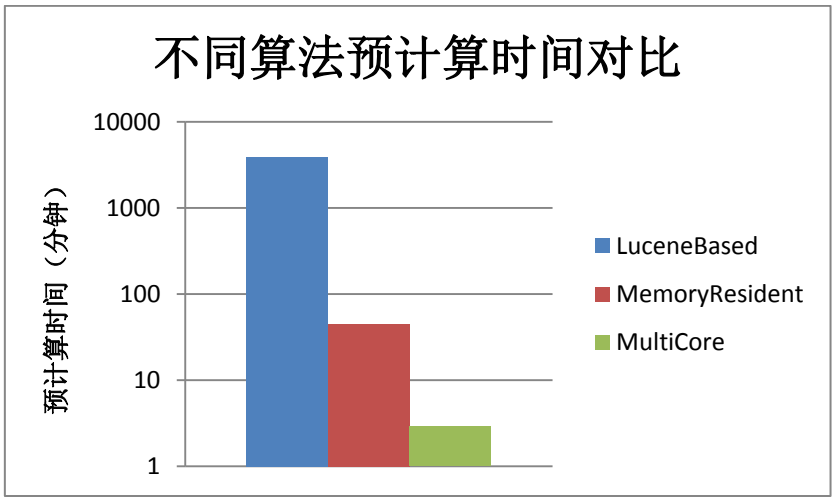

图 7.3 不同算法预计算时间对比

7.4.2 不同方法占用内存比较

从下图可以看出 MemoryResident 方法和 MultiCore 方法都会占用大约 9.2G 左右的内存空间，这是因为它们在预计算的时候都将一阶单词相似度矩阵读入了内存里。相反，LuceneBased 方法将一阶单词相似度矩阵存放在了磁盘上的 Lucene 索引里，需要用到数据的时候才从磁盘读入，因此占用内存非常小。

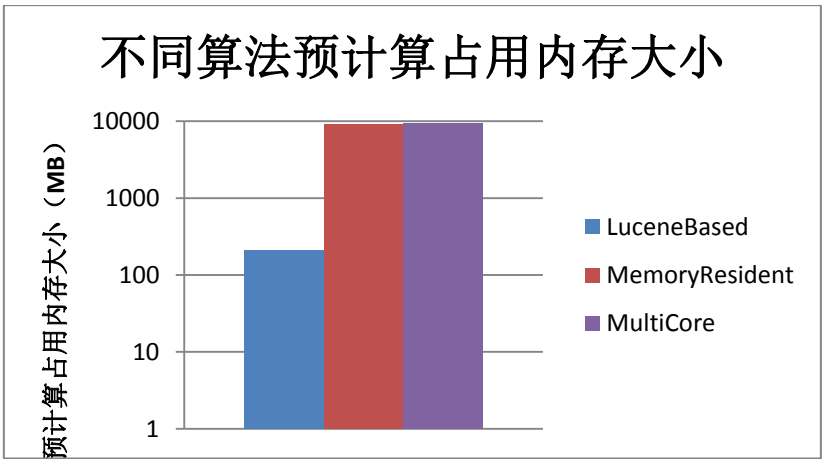

图 7.4 不同算法预计算占用内存大小

7.4.3 内存使用量对多核并行计算方法执行效率的影响

为了考察在内存容量有限而不能将一阶单词相似度矩阵完全载入内存的情况下多核并行算法预计算的效率，我们通过在内存中建立一个一阶单词向量的哈希表和相应 LRU 队列的方法（详见第五章“单词相似度计算”）使得各个并行计

算单元之间共享某些相同的一阶单词向量来同时达到减少内存使用量和减少磁盘操作的目的。哈希表的大小根据内存容量大小做出相应调整，图（7.5）给出了不同内存容量限制下多核并行计算方法预计算消耗的时间。可以看出随着内存容量的增大，预计算时间几乎呈现线性减小的趋势，这是因为此时哈希表的大小也线性增长，并行计算单元之间共享的概率也线性增加，磁盘访问量也随之线性减少，当内存大到能容纳整个一阶单词矩阵时每个单词对应的一阶单词向量最多只会从磁盘读入一次，消耗时间大大减少。

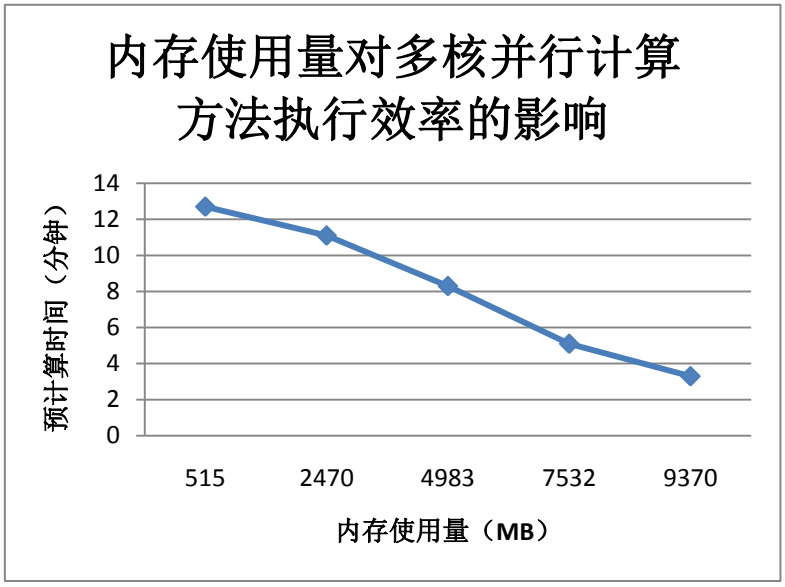

图 7.5 内存使用量对多核并行计算方法效率的影响

7.5 二阶单词相似度矩阵效果测试

本小结实验将考察 LSA、PMI-IR 两种方法计算出的单词相似度和二阶单词相似度（取自计算出的二阶单词相似度矩阵）的效果。在对比相似度计算效果的时候我们把 WordNet::Similarity 计算出的单词相似度当作标准，将另外三种相似度计算方法计算出的单词相似度与之对比，考察各自方法与其结果的相关性大小，相关性越大的方法说明其效果越好。

由于 WordNet::Similarity 只能计算名词之间的相似度而不能计算形容词之间或者副词之间的相似度，而且不能计算不同词性单词之间的相似度，所以我们的测试集中单词必须都是名词。Finkelstein et al. (2001)[34]中准备了 353 对（名词-名

词）单词对研究单词相似度计算的效果，这里我们也使用这 353 个名词对来考察涉及到的三种单词相似度计算效果。这 353 个名词对中有 7 个名词对中至少包含了一个 WordNet 中不曾出现过的单词，所以最终我们只选取了剩下的 346 个名词对作为测试单词对。

首先我们使用包括 WordNet::Similarity 在内的四种方法对这 346 个单词对计算出相应的单词相似度，这样每种方法会计算得出一个长度为 346 的向量，向量里的第 i 个值对应了第 i 个单词对之间的相似度。我们将 LSA、PMI-IR 和二阶单词相似度计算方法得出的向量分别和 WordNet::Similarity 方法得出的向量计算关联性，关联性越大的说明效果越好。关联性的计算方法采用 Pearson-Correlation，公式如下所示，其中 X 和 Y 是我们要计算关联性的两个相似度向量，n 是 X 和 Y 的长度，这里等于 346：

$$\frac{\sum_{i=1}^{n}(X_i-\overline{X})(Y_i-\overline{Y})}{(\sqrt{\sum_{i=1}^{n}(X_i-\overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\overline{Y})^2})}$$

对比实验中使用的 LSA 算法的实现取自 http://lsa.colorado.edu。

PMI-IR 算法实现来自 Turney（2001）[35]。该文章中的方法能通过查询 AltaVista 搜索引擎来计算任意两个单词之间的相似度，计算公式如下，是最基本的 PMI 计算方法的变种：

$$PMI-IR(w_1,w_2)=\log(\frac{H(w_1 Near w_2)}{H(w_1)H(w_2)})$$

上式中H(w)是在 AltaVista 上查询w时的返回结果数。从公式可以看出如果两个单词经常在同一个网页中同时出现地很近的话两个单词的相似度就会很高。

关于 WordNet::Similarity，有 python 语言的模块[11]和 perl 语言的模块[12]。它们提供了 WordNet 里六种单词相似度的计算方法，分别是 lch[36]、wup [37]、path、res[38]、lin[39]还有 jcn[40]。

Lch 方法算出两个单词所在概念节点之间最短路径长度，然后除以它们在 is-a

[11] http://nltk.googlecode.com/ svn/trunk/doc/howto/wordnet.html
[12] http://qwone.com/~jason /WordNet/

层次图中的最长路径长度。Wup 方法计算两个单词所在概念节点之间到达最近公共概念祖先 LCS（Least Common Subsumer）之间的路径长度，然后除以两个单词所在概念到达顶端祖先概念节点的长度和。Path 方法计算两个单词所在概念节点之间的长度的倒数。剩下的其他三种方法均是根据信息量 IC（Information Content）来计算的。

我们将 WordNet::Similarity 六种相似度计算方法计算的相似度向量作为标准结果，然后将其他方法计算所得的相似度向量和其计算关联性，得到了表（7.2）中的结果：

表 7.2 不同方法计算结果和 WordNet::Similarity 计算结果的关联性

	Jcn	Lch	Lin	Path	Res	Wup	Avg
PMI-IR	0.14	0.12	0.06	0.15	0.22	0.11	0.13
LSA	0.16	0.26	0.21	0.29	0.28	0.22	0.24
一阶相似度	0.15	0.40	0.39	0.35	0.44	0.40	0.36
二阶相似度	0.38	0.39	0.33	0.45	0.42	0.33	0.38

从上表平均结果可以看出，PMI-I 结果最差，而二阶相似度计算出的结果 Pearson Correlation 最高。虽然 LSA 方法效果较好，但是仍然和一阶相似度和二阶相似度计算结果有较大差距。我们得出的结论是二阶单词相似度计算方法的效果最好。

如果两个单词都是对方的最相似单词，则这两个单词被称为互为最相近单词（Respective Nearest Neighbor, RNNs）。对于在维基百科单词集中至少出现了 50 次的 5230 名词中，本文计算发现了一共 622 对 RNN 名词。下表给出了其中大概 25 分之一的的 RNN，而且每隔 10 个取一个展示。

表 7.3 RNN 单词对

序号	单词 1	单词 2	相似度
1	earnings	profit	0.50
11	revenue	sale	0.39
21	acquisition	merger	0.34
31	attorney	lawyer	0.32
41	data	information	0.30
51	amount	number	0.27
61	downturn	slump	0.26
71	there	way	0.24
81	fear	worry	0.23
91	jacket	shirt	0.22
101	film	movie	0.21
111	felony	misdemeanor	0.21
121	importance	significance	0.20
131	reaction	response	0.19
141	heroin	marijuana	0.19
151	champions	tournament	0.18
161	consequen	implication	0.18
171	rape	robbery	0.17
181	dinner	lunch	0.17
191	turmoil	upheaval	0.17
201	biggest	largest	0.17
211	blaze	fire	0.16
221	captive	westerner	0.16
231	imprisonm	probation	0.16
241	apparel	clothing	0.15
251	comment	elaboration	0.15

可以看出所给例子中的效果都不错。只有几个看起来好像不对，第一眼看上去好像不大可能。但只要仔细分析以下就可以看出来它们为什么比较相似了。比如标号为 221 的单词对“captive”和“westerner”。不管怎么想，任何人工编辑的词典里也绝不会把这两个单词看作相似词。实际上在 274 次“westerner”出现的语境里，有 55%的情况中是指“westerner in captivity”。

7.6 地理主题模型参数预计算测试

本小节主要对第六章提出的基于地理主题模型的推荐算法中的模型参数估计预计算进行考察，主要讨论预计算所需要消耗的 CPU 时间。我们通过调整各种

参数来量化它们对计算时间的影响，参数说明请见表（7.4），其中取值范围中粗体表示的数字是实验时的默认参数值。在每组实验中我们调整其中一种参数，固定所有其他参数，考察该参数对计算时间的影响。在下面的实验中 CPU 所耗时间是指 EM 算法中每次迭代所消耗的时间。

表 7.4 实验参数取值范围

参数	取值范围
照片数量 D	9339,11863,16106,24665,**50094**
主题数量 K	**3**,11,21,31,41,51
区域数量 N	**10**,20,30,40,50

需要说明的是，在实验的时候我们只选取了照片集合中出现次数超过 100 次的标签作为单词集，所以随着照片集大小的变化单词集也会变化。

7.6.1 照片数量 D 的影响

图（7.6）反应了照片数量的多少对每次迭代消耗的时间的影响。可以看出随着照片数量的增加，计算开销逐渐增大。从公式（6.15）中可以看出每次迭代消耗时间随着 D、W、V 的增加而增加，这里照片数量的增大会同时使得 W 和 V 的大小同时变大，参见图（7.6），所以照片数量的增加会显著增加计算时间。

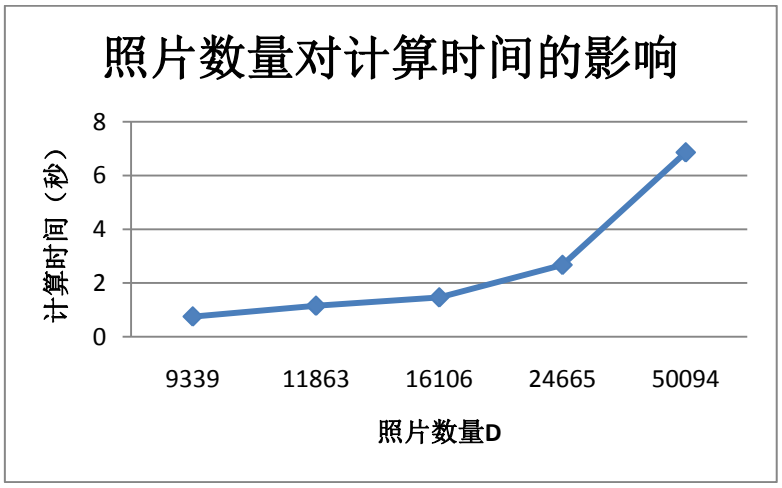

图 7.6 照片数量 D 对计算时间的影响

图（7.7）说明随着照片数量的增加不同标签数量 V 大小显著增加，另外总标签数量 W 当然也增加了。

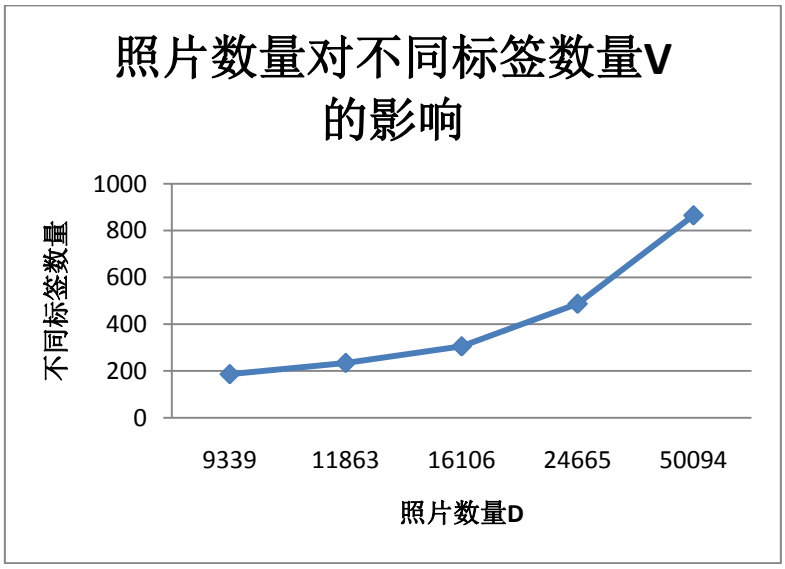

图 7.7 照片数量 D 对不同标签数 V 的影响

7.6.2 不同主题数量 K 的影响

图（7.8）反应了主题数量对计算时间的影响。可以看出随着主题数量的增加计算时间在慢慢上升，但是没有照片数量的影响巨大。这可以从公式（6.15）中可看出，虽然公式（6.15）说明计算时间也和 K 同时增大，但是主要还是受控于 W 和 D 的影响，而 W 和 D 的大小又远大于 K 的大小，K 只能控制KNV和T_2KNV两项，所以 K 的增大只能较小地使得计算时间增大。

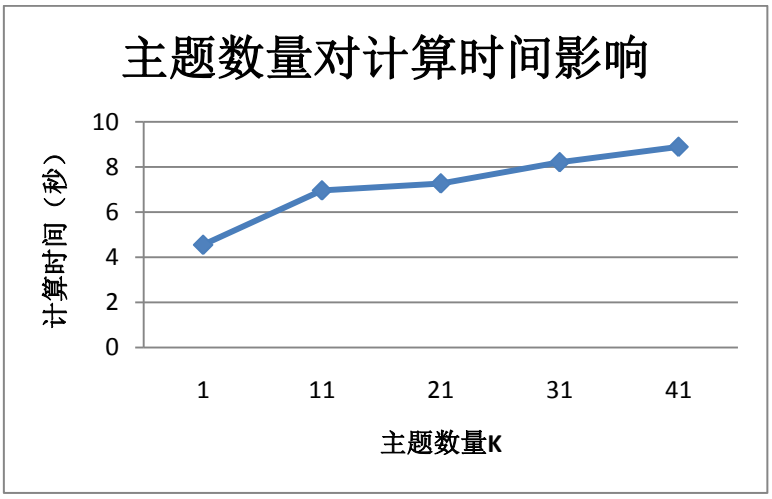

图 7.8 主题数量 K 对计算时间的影响

7.6.3 区域数量 N 的影响

图（7.9）反应了区域数量对计算时间的影响。从图中可以看出计算时间的多少几乎和区域数量成正比关系。这其实从公式（6.15）中比较容易看出，因为公式（6.15）中的每一项都包含了因子 N，所以随着 N 的增大，计算开销线性增加。

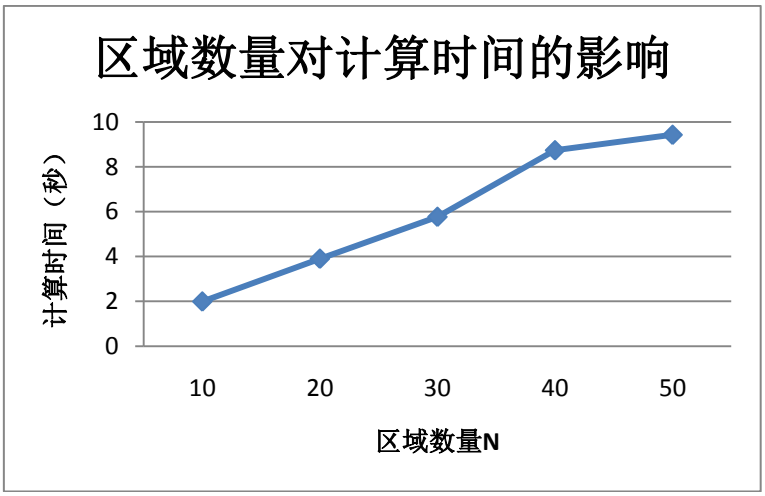

图 7.9 区域数量 N 对计算时间的影响

7.7 本章小结

本章说明了实验数据集的获取方法，包括照片元数据、音乐元数据、音乐内容数据以及维基百科语料库，并展示了已经实现的 Pictune 原型系统，然后根据实验数据分析了不同单词相似度预计算方法的性能以及单词相似度计算的效果，

最后讨论了参数的变动对推荐算法预计算时间的影响。

第8章 总结和展望

8.1 本文主要工作和贡献

本文对基于移动用户地理信息的音乐推荐进行了深入的研究，主要工作和贡献如下：

（1） 提出了一种非常实用的音乐推荐应用。

基于移动用户地理信息的音乐推荐。由于手持移动终端数量的爆炸式增长，移动用户的数量也几何级增加，传统的音乐推荐场景已经不能满足移动用户的需求，本文适时提出了这一种移动环境中的推荐服务，它能根据用户所处环境的不同、心情感受的不同推荐最适宜的音乐，具有非常实用的价值。

（2） 设计并实现了一种可行的基于地理信息的音乐推荐系统原型。

针对移动环境中的音乐推荐特点，我们抓取了大量的带有 GPS 坐标的照片元数据、音乐元数据、音乐内容数据，并高效地用空间索引将这些数据组织了起来，而且用本文中提出的单词相似度计算模块与推荐算法模块中的预处理方法进行了高效地预处理，形成的系统原型支持实时的移动端音乐推荐请求。

（3） 设计了一种高效易用的任意单词相似度计算方法。

在前人工作的基础上设计了一种简单易用的单词相似度计算方法，并提出了基于 Lucene 索引的实现方法、常驻内存的实现方法、多核并行计算的实现方法，最终得到了一个单词相似度矩阵，支持任意英文单词相似度之间的计算。

（4） 设计了一种高效的音乐推荐算法

对照片数据集建立了地理主题模型，使用 EM 算法对分块后的每个区间的照片数据进行参数估计，最后在实时推荐音乐的时候首先找到用

户所在的区间，取出模型参数，然后再扫描音乐集找到最匹配的 Top-K 首音乐。将大部分的计算复杂度放在了预计算里面，从而大大降低了在线推荐算法的复杂度。

（5） 通过实验验证了整个推荐系统的有效性和高效性

通过在真实数据集上的实验证明了单词相似度计算算法和推荐算法的高效性，并验证了单词相似度计算的效果，并给出了 Pictune 运行的实例。

8.2 未来研究工作展望

虽然本文解决了大部分的问题，但是在未来仍然有以下重要的问题可以研究：

（1） 数据更新问题。现在对地理主题模型的参数进行估计的时候是需要预计算的，需要同时知道所有照片的数据，但是我们的后台爬虫程序一直在爬取数据，现在的做法是每隔一段时间定期重新计算一次模型参数，这个方法效率较低，在将来重点考虑如何设计增量式更新的算法。

（2） 中文音乐推荐。Pictune 现在只能推荐英文的音乐，如果要能推荐中文音乐，有诸多的问题需要解决，比如中文数据获取、中文分词、中文单词相似度计算等等棘手问题。

（3） 推荐质量的量化与用户反馈。Pictune 现在对音乐推荐质量的衡量仅仅是通过用户打分来反应的，未来的工作将着重研究如何量化推荐质量以及如何根据用户的反馈来不断改进推荐质量。

（4） 融合个性化推荐方法。Pictune 现在仅仅考虑了用户所在的位置信息和心情信息，没有考虑用户个性化的信息，比如用户的年龄、性别、爱好等内容，如何在现有推荐算法中融合进用户的个性化信息是一个重要问题。

参考文献

[1] G.Turkiyyah. Foundations of multidimensional and metric data structures[M]. USA:Morgan Kaufmann,2008,40(4):518-519

[2] A.Guttman. R-trees: A dynamic index structure for spatial searching[C]. SIGMOD,1984:47-57

[3] T.K.Sellis, N.Roussopoulos, C.Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects[C]. VLDB,1987:507-518

[4] N.Beckmann, H.-P.Kriegel, R.Schneider, et al. The r*-tree: An efficient and robust access method for points and rectangles[C]. SIGMOD,1990:322-331

[5] D.Hilbert. Ueber stetige abbildung einer linie auf ein flachenstuck[C]. Math-ematische Annalen,1891:459-460

[6] G.R.Hjaltason, H.Samet. Distance browsing in spatial databases[J]. ACM Trans. Database Syst.,1999,24(2):265–318

[7] Zhexuan Song, Nick Roussopoulos: K-Nearest Neighbor Search for Moving Query Point[C]. SSTD 2001:79-96

[8] Sarana Nutanong, Rui Zhang, Egemen Tanin, Lars Kulik: The V*-Diagram: a query-dependent approach to moving KNN queries[J]. PVLDB 1(1):1095-1106 (2008)

[9] oseph O'Rourke. Computational Geometry in C. Cambridge University Press. 2001

[10] Rich, E.(1979). User modeling via stereotypes. In Cognitive Science: A Multidisciplinary Journal, volume Vol. 3, No. 4, pages 329–354

[11] Shardanand,U. Social information filtering for music recommendation. Master's thesis, Massachussets Institute of Technology. 1994

[12] Salton, G. and McGill, M.J. Introduction to Modern Information Retrieval.McGraw-Hill, Inc., New York, NY, USA. 1986

[13] Joseph O'Rourke. Computational Geometry in C. Cambridge University Press. 2001

[14] Ae-Ttie Ji, Cheol Yeon, Heung-Nam Kim, GeunSik Jo: Collaborative Tagging in Recommender Systems[C]. Australian Conference on Artificial Intelligence 2007:377-386

[15] Tso-Sutter, K.H.L., Marinho, L.B., and Schmidt-Thieme, L. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In Proceedings of the 2008 ACM symposium on Applied computing, pages 1995–1999, New York, NY, USA. ACM. 2008

[16] Mark Levy, Mark Sandler: A Semantic Space for Music Derived from Social Tags[C]. ISMIR 2007:411-416

[17] Marius Kaminskas, Francesco Ricci. Location-Adapted Music Recommendation Using Tags. In Proceedings of UMAP'2011. pp.183-194

[18] P.Dunker, S.Nowak, A.Begau, C.Lanz, Proceeding of the 1st ACM international conference on Multimedia information retrieval MIR 08 , 97(2008).

[19] D.Yang and W.Lee. Disambiguating music emotion using software agents. In Proc. of the Int. Conf. on Music Information Retrieval, 2004.

[20] W.Wei-ning, Y.Ying-lin, and J.Sheng-ming. Image retrieval by emotional semantics: A study of emotional space and feature extraction. IEEE Int.Conf. Systems, Man and Cybernetics, 4, 2006.

[21] Turnbull, Douglas, Luke Barrington, and Gert Lanckriet. Five approaches to collecting tags for music. Knowledge Creation Diffusion Utilization (2008) : 225-230.

[22] Landauer, T.K. and S.T.Dumais. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. Psychological Review 1997 104(2):211–240.

[23] Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi: WordNet: : Similarity - Measuring the Relatedness of Concepts[C]. AAAI 2004:1024-1025

[24] P.Kolb. Disco: A multilingual database of distributionally similar words. In Proc. KONVENS, 2008.

[25] David M. Blei, Andrew Y. Ng, Michael I. Jordan: Latent Dirichlet Allocation. Journal of Machine Learning Research (JMLR) 3:993-1022 (2003)

[26] Aho, Alfred V., Margaret J.Corasick. Efficient string matching: An aid to bibliographic search. Communications of the ACM 18 1975(6): 333–340

[27] Zelig S.Harris. Mathematical Structures of Language. Wiley, New York. 1968

[28] Hiyan Alshawi, David M. Carter: Training and Scaling Preference Functions for Disambiguation. Computational Linguistics (COLING) 20(4):635-648 (1994)

[29] Christopher M. Bishop, Pattern Recognition And Machine Learning, 2007, Springer

[30] Hiyan Alshawi and David Carter. Training and scaling preference functions for disambiguation. Computational Linguistics, 1994 20(4):635–648, December.

[31] Dekang Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In Proceedings of ACL/EACL-97, 1997, pages 64–71, Madrid, Spain, July.

[32] Dekang Lin: Automatic Retrieval and Clustering of Similar Words. COLING-ACL 1998:768-774

[33] C.Zhai, A.Velivelli, and B.Yu. A cross-collection mixture model for comparative text mining. In KDD, pages 743–748, 2004.

[34] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, Eytan Ruppin: Placing search in context: the concept revisited. WWW 2001:406-414

[35] Peter D. Turney: Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. ECML 2001:491-502

[36] Leacock, C., and Chodorow, M. Combining local context and WordNet similarity for word sense identification. In Fellbaum, C., ed., WordNet: An electronic lexical database. MIT Press. 1998. 265–283.

[37] Wu, Z., and Palmer, M. Verb semantics and lexical selection. In 32nd Annual Meeting of the Association for Computational Linguistics, 1994. 133–138.

[38] Resnik, P. Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995. 448–453.

[39] Lin, D. An information-theoretic definition of similarity. In Proceedings of the

International Conference on Machine Learning. 1998.

[40] Jiang, J., and Conrath, D. Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings on International Conference on Research in Computational Linguistics, 1997. 19–33.

攻读硕士学位期间主要的研究成果

[1] Ke Chen, Gang Chen, Lidan Shou, Fei Xia: Pictune: situational music recommendation from geotagged pictures. SIGIR 2012:1011

致谢

两年半的研究生生涯即将结束，值此毕业之际，我非常感谢这段时间以来我的导师陈刚教授、寿黎但副教授、陈珂副教授对我在科研学习上的孜孜教导和生活上无微不至的关心。他们渊博的知识、治学的严谨让我受益匪浅。从他们身上我学会了科研的方法，少走了很多弯路，他们对我的教导将使我受益终身。

感谢浙江大学数据库实验室的周显科锞博士、王振华博士、朱珠博士、曹晖博士、徐昶博士、蔡华林博士、柏壑博士、尚璇博士、马春阳博士、张晓龙博士、洪银杰博士、陈楠博士，在平时的生活上对我的关心帮助，他们学术上的造诣和工程上的技术给予了我莫大的帮助。

感谢实验室的胡炜、郑山、张贤、周健、马逸、陈树杰、张栋、庞贵峰、马伟师兄师姐在我读研期间对我的帮助，你们教会了我很多的技术和做事的方法，使我受益颇多。

感谢同届的张超、黄攀、毛旷、骆歆远、汪荣、徐志荣、娄超、邵海东、刘博、张鹏、戴丹、王昱，有了你们我的研究生生活更为充实，留下了一些美好的回忆。

还要感谢张冰冰、刘云飞、彭湃、史立彬、王俊俏等师弟师妹，与你们在一起工作的过程中同样获益良多。

感谢我的父母，对我二十多年来的养育之恩，感谢你们对我决定的支持，给予了我非常温馨的家，衷心祝愿你们身体健康，生活快乐。

最后，感谢评阅、评议论文和答辩委员会的各位专家学者在百忙的工作中能给予指导。

夏飞

2013年1月于浙大求是园