## 内容热度增长与消退

### 1.1热度的定义与目的

一般定义下，热度是指某一事物在一定的时间区间中受关注与谈论的程度，不同的事物的有不同的量化方式。热度通常表现在数量和质量方面上。数量方面，热度可以通过浏览量、点击量、转发量等指标衡量；质量方面，则涉及到用户的参与行为，如点赞、评论、收藏。

定义之前必须要明确目的。在软件系统中设计热度本质上是为了产生更多的收益。对于不同事物的具体目的有差异。比如，对于文本内容来说，更大的热度能有更好的传播效果并产生更多的收益；对于话题来说，高热度则代表着高关注、高讨论；而对于事件而言，更大的热度可能会带来更大的影响力。

在主题社区系统中，社区中的内容是热度表现的客体，包括文本内容、视频内容、话题内容等内容。在本设计中，文本内容是本设计中的主要形式，也是主要的处理对象。热度的设计可以带来多方面的好处，包括：

1. 增强用户参与度：针对宠物主题设计的热度机制可以吸引更多的用户参与、分享和评论，从而增强用户参与度。
2. 增强内容曝光度：高热度代表内容能被更多用户获知，也就是内容获得更大程度的曝光。
3. 促进平台流量增长：热门内容和话题往往可以吸引更多用户访问和使用平台，从而促进平台的流量增长和用户规模扩大。

上述几点好处就是本系统设计热度的重要目的。综上所述，本系统将热度定义为文本的

内容的点击量、转发量、点赞数量、评论数量、收藏数量在一定的数学模型下的量化数据；在系统中设计热度的目的是为了增强用户参与度、增强内容曝光度、和促进平台流量增长。热度定义中所提及的数学模型是热度量化结果好坏的关键，相关内容将在下文中介绍。

### 1.2热度增长与消退的数学原理

热度必须是变化的，而且一个内容或者事件不能长时间处于高热度或者是低热度的状态，也就是说热度模型的设计必须包含热度的增长和热度的消退。高热度内容需要随着函数因子以一定的速度消退，低热度的内容则需要一定的刺激函数触发其热度的增长。

在选取合适的模型拟合之前，需要先明确事件的生命周期。以文章为例，其生命周期主要有：文章创建阶段、热度增长阶段、热度饱和阶段、热度消退阶段和稳定阶段。不同的生命周期具有转换关系，具体如下。【做个标记todo】

热度增长与消退的数学原理可以通过多种模型进行解释和预测，其中常见的有SIR模型、Gompertz曲线和Logistical曲线【做个标记todo】。

【做个标记todo】文章热度变化与视频的热度变化有很大的相似点。针对视频网站的调研有利于发现背后是数学奥秘。

这些模型偏向于对事件热度变化的解释和预测，对于本设计有重要的参考价值。在参考多种模型的情况下，很容易得出热度变化强烈依赖时间这个参数，其他的参数则因不同的事件而有差异。针对上文的定义，本系统影响文章内容的热度主要参数为：时间、点击量、转发量、点赞数量、评论数量和收藏数量。参数与结果建的对应关系为：

hots = (Math.log10(article.getViews()) + article.getLikes() + article.getFavorites() + article.getComments()) / age; 【做个标记todo】

【做个标记todo】画图

## 内容推荐

内容推荐是以内容输出为主的网站或系统维持用户数量、系统活跃的核心内容。随着用户需求和个性化呈现的不断提高，基于协同过滤、深度学习、自然语言处理等技术的推荐算法已经成为很多网站和应用的核心功能。通过分析用户历史行为、兴趣和偏好，推荐算法可以为用户提供更加个性化和精准的内容，从而提高用户留存率和忠诚度。同时，推荐算法也可以优化内容生态和商业模式，增加收入来源和利润空间。因此，内容推荐已经成为现代互联网生态系统中不可或缺的一部分。

### 2.1基于文本特征的内容推荐

自然语言处理技术的发展为文本特征的提取提供了可能。提取文本的特征将有多种好处，如文本分类、情感分析、文本生成等。本系统使用文本特征提取技术的主要目标是实现内容推荐，也就是说根据文章的特征，给用户推荐相似特征的文章。

基于文本特征的内容推荐是一种利用自然语言处理技术，从用户历史行为和物品信息中挖掘关键词、主题和情感等文本特征，通过计算相似度来预测用户可能感兴趣的内容，并向其推荐相关的物品或信息。具体来说，该算法会对用户历史行为和物品信息进行文本分析，提取关键词、主题和情感等特征，并根据这些特征计算用户与物品之间的相似度，从而推荐最符合用户需求的内容。基于文本特征的内容推荐可以应用于各种领域，如搜索引擎、社交媒体、新闻资讯等，已经成为提高用户满意度和参与度的重要推荐手段。【做个标记todo】

为了更好的实现内容推荐，本系统将推荐过程分成了三个阶段：特征提取、相似度计算和用户导向。

#### 2.1.1特征提取。

特征提取就是提取文章的特征信息。本系统使用了TF-IDF模型（Term Frequency-Inverse Document Frequency Model），统计每个单词在文章中出现的次数，去除特征无关的单词，并计算单词在参考语料库中出现的频率，根据TF-IDF的若干关键词作为文章的特征。

TF-IDF是一种常用的文本特征权重计算方法，它可以帮助我们评估一个词语在一个文档中的重要程度。TF-IDF的值越高，表示该词语在文档中越重要。一个词语在一个文档中的TF-IDF值可以通过以下公式计算：

TF-IDF = TF \* log(N / DF)

其中，TF表示该词语在文档中出现的次数（词频），DF表示该词语在所有文档中出现的文档频率，N表示文档总数。当一个词语在所有文档中都出现时，DF的值为N，log(N/DF)的值为0，表示该词语对于区分不同文档的重要性为0。

在当前技术的支持下，很容易实现文本分词和词频统计。为了使得词频统计更加合理，在统计前需要去除特征无关的单词，如常见的介词、副词。系统选用了TF-IDF模型，词频统计过程中必须要考虑单词在整个语料库中出现的频率，以此捕捉单词的重要性，减少一些常见词汇的影响。同时必须要注意到，在一个系统中，语料库即系统的文章数量的总和极其庞大，并且一直处于变化状态。语料库的选取会很大程度地影响TF-IDF值的计算值，本系统并不使用数据库中的全部文章数据集作为语料库，算法计算时选取相似标签的文章和相同类别的文章作为语料库。这样做的好处是提高算法效率，同时可以提高特征提取的质量。

#### 2.1.2相似度计算

相似度计算是内容推荐的重点，本质上是计算两个特征向量的余弦相似度。假设有A和B两个向量，它们的维度相同，则它们的余弦相似度为：

余弦相似度的取值范围为[-1,1]，值越接近1则表示两个向量越相似，值越接近-1则表示两个向量越不相似，值为0则表示两个向量完全无关。

计算两个特征向量的余弦相似度最重要的是要保证两个特征向量的维度相同。在特征提取时，文章的若干关键词被作为了该文章的特征，也就是不同文章的特征不尽相同，文章的特征向量之间的维度有差异。在计算相似度之前，需要对两个向量在维度上进行统一。

统一两个特征向量的维度的一个简单的方法是对两个特征向量的属性进行合并。简单来说，对于向量A和向量B，将向量B的属性合并到向量A中，向量B存在但向量A不存在的属性值填充零值，两者都存在的属性无需处理。

另外一个方法是构造出一个公共特征向量。通过聚类的方式可以构造出这样的向量，如k-means聚类算法。构造公共特征向量一方面可以计算文章之间的相似度，同时还可以实现对文章的分类。

上述两个方法中，对向量的属性进行合并可以使两个特征向量的相似度计算更依赖于各自的属性，假设两者的余弦相似度较为接近1，则两篇文章必定是特征相似的。同时，它也有一定的缺点，比如两个属性相似但不相同，则它们会被视作是两个独立的属性，也就是放大了两者之间的差别，最终会结果的可信度下降。使用聚类的方式计算时需要不断改变公共特征向量的属性（在聚类中，公共特征向量通常被称作蔟中心），聚类结果的各个类别中，会有包含很多离蔟中心相对较远的数据。简单来说，聚类的结果中，文章之间的相似度可能不会很高。总之，两种方式各有优略，合并向量的属性的方法适用在计算相似度强的场景，聚类方式则顾名思义，更适合对若干文章进行划分。

### 2.2基于用户反馈的内容推荐

基于文本特征的内容推荐能够很好的给用户推荐内容，同时也应该注意到两方面的问题：基于自然语言处理技术提取的文本特征不一定能够很好的表示文本的特点；一些文本不一定具有显著特征。这两种问题都会不同程度的影响推荐的效果，系统需要一种不依赖于文本特征的推荐算法实现推荐功能。

基于用户反馈的内容推荐是指根据用户过去的行为和偏好，提供个性化的内容推荐，是一种更加个性化和精准的推荐方式。与基于用户偏好的推荐不同，它会根据用户对先前推荐内容的反馈（如点击、收藏、评论等）来调整推荐策略。实现基于用户反馈的内容推荐需要考虑到以下几个方面问题：

1. 用户反馈数据的收集和存储：需要通过技术手段收集用户反馈数据，并将其存储在数据库中，以备后续分析和使用。
2. 反馈数据的处理和分析：需要使用机器学习和数据挖掘等技术，对用户反馈数据进行处理和分析，提取出用户对不同内容的偏好及关注度等信息。
3. 推荐算法的设计和实现：需要根据用户反馈数据，设计合适的推荐算法，以便更加精准地推荐符合用户兴趣和需求的内容。
4. 实时性的保证：需要实时地更新和调整推荐策略，以应对用户行为的变化和新内容的出现。

#### 2.2.1 历史记录的筛选

用户不希望收到已点击的文章，需要历史记录来帮助排除已经浏览过的内容。一个用户的浏览记录数据对于该用户来说已经没有挖掘的价值，但是对另外的用户却不一定。事实上，一些用户的浏览记录很可能成为一部分用户的潜在浏览对象。简单来说，筛选部分用户的历史记录，整理这些历史记录之后重新推荐给另一部分用户。

筛选历史记录可以考虑很多条件，其中时间范围和用户点击频率是重要的两点。最新发布的文章（如最近3天）由于热度还没增长，点击量不会很高，这段时间的文章内容可能具备被推荐的潜力。文章内容对于用户来说像是一次性消耗品，短时间内用户往往不会反复点击同样的内容，如果存在一篇文章在较短时间内被多次访问，则能说明这篇文章是值得反复观看的，同时也是值得被推荐的。

#### 2.2.2 基于历史记录的用户标签与内容推荐

在前面的处理中已经能够很好的提取文章的特征，这些特征一方面能够代表文章的特点，同时也可以用户个性的象征。相对于基于用户信息的用户标签，基于用户的历史记录的用户标签能够更好的表示用户在系统当中的爱好情况，系统则可以根据这个用户标签给予用户更加个性化的推荐。

将用户浏览记录的文章标签作为该用户的用户标签，基于用户标签的系统推荐作为驱动数据，监听用户反馈信息，把正反馈的文章特征添加到用户标签数据中。如此反复便可实现用户标签的更新与内容推荐。需要注意两点，用户的标签必须是有限的，这样才能保证推荐的相关性；当用户标签的长度到达其限度时，更新用户标签要满足先进新出原则，这是保证推荐的实时性的关键。

在用户标签的更新与内容推荐的过程中，如果该用户对于不具有显著特征的文章给予了正反馈，则可证明当前的用户标签与文章的实际特征有关联，可以将用户的标签属性赋予特征向量。经过多个用户的多次修正，最终可以构造出有意义的特征向量。同样的方式也适用于普遍情况，文章的特征是可以根据用户反馈使其特征具有代表性的。

## 3 用户关系与用户发现

### 3.1用户关系的表现形式

在系统设计中，用户关系可以有多种表现形式，具体取决于不同的场景和应用。一个合理的用户关系的设计可以提高用户获取内容的效率与质量。在本系统中，用户关系的主要表现形式是关注/粉丝。用户可以通过关注其他用户来建立关系。被关注的用户成为粉丝，可以看到关注者的动态，获取相关内容的更新。

用户关系的建立是一个双向的过程，既需要用户的主动参与，也需要系统的介入和引导。系统通过特定的算法向用户推荐相关的用户或内容，帮助用户发现更多有价值的资源。同时，系统提供良好的用户体验和界面设计，使用户能够轻松地建立关系、查看动态和获取更新。在用户和系统的共同努力下，实现一个高效、优质的用户关系网络。本设计把用户关系中系统的介入过程定义为用户发现。简单来说，系统通过特定的算法向用户推荐用户，再由用户完成关系的建立。

### 3.2基于关联性分析的用户发现

尿布与啤酒的故事早已耳熟能详。事实上，通过大量的数据分析和用户行为分析，人们很容易发现消费者对于不同种类商品行为的关联性。但是当数据量足够大，比如数量级为亿级别的时候，传统的分析方法可能会失效，或者说数据量巨大使得人们无法通过传统的运算找出不同行为之间的关联性。

计算机辅助目前的最优方案。为了让设计的系统能更好的工作，需要给系统设计一个好的算法，比如经典的关联性分析算法：Apriori算法和FP-growth算法。

Apriori算法是一种常用的数据关联性分析挖掘算法，其根本原理是找出数据集中频繁出现的数据集合。关联规则是形如X→Y的表达式，使用支持度（support）和置信度（confidence）来度量。其中：

支持度：

置信度：

例如，对于如下规则：

{尿布}→{啤酒}

假设{尿布,啤酒}的支持度为3/5，{尿布}的支持度为4/5，则{尿布}→{啤酒}的可信度为3/4，其解释为，购买了尿布的顾客都会购买啤酒，这条规则的可信度有75%。

可以说，Apriori算法是基于统计的，根据这个算法的挖掘关联规则的集体过程是先统计出频繁项集，然后根据频繁项集找出强关联规则。不难发现，Apriori算法是需要人为定义频繁项集的，比如定义出现频率超过70%的集合为频繁项集。另外，上述的公式还蕴含了众所周知的朴素道理，如果一个事件出现的概率足够高，那么该事件对于其他事件的推导的可信度就越低，比如，广东夏天白天气温高于25℃，这是一个概率极高的事件，对于另外的事件，如“出门逛街”，那么气温高于25℃推导出“出门逛街”的置信度就不会很高，类似于这种推导是没有意义的。

Apriori算法的中间产物是频繁项集的关联树，找出强关联规则的方法就是遍历搜索。事实上，想要尝试暴力搜索几乎是不可能的，尤其是对于有上亿规模的数据集。前人已经帮我们总结出规律：如果一个集合是频繁项集，则它的所有子集都是频繁项集；如果一个集合不是频繁项集，则它的所有超集都不是频繁项集。利用这两个规则对关联树进行剪纸，可以极大加快搜索速度。

FP-growth算法是基于Apriori算法构建出来的，它采用了名为频繁模式树的高级数据结构使得扫描次数减少，从而大大加快了算法速度。FP-growth算法只需要对数据库进行两次扫描，而Apriori算法对于每个潜在的频繁项集都会扫描数据集判定给定模式是否为频繁项集，因此FP-growth算法的速度要比Apriori算法快。

FP-growth算法的成功很大程度是FP-tree的功劳。FP-tree是一种前缀树，所谓前缀树，即当前节点和后续节点都拥有共同前缀。这跟Apriori算法的树的增长类似。FP-growth算法的优秀之处在于，它首先有频繁项集的重写过程，根据频繁项集和频繁项集中的项频率进行排序，然后根据先后顺序构建FP-tree。FP-tree中的每一个节点，基本的数据结构是字典：节点保存自身信息和频率，构建树时每次遍历到该节点则频率加一，若不存在该节点则新增该节点。此外，同一事件可能在树中多次出现（前缀不一样），因此还需要额外的指针指向相同的事件，以便遍历。下图是经典的FP-tree。

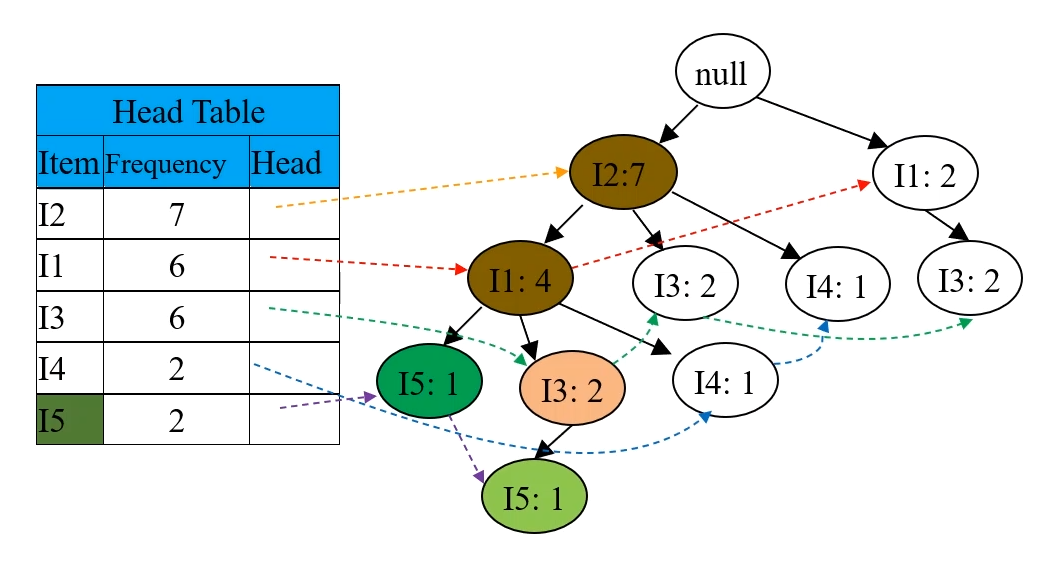


图3.2.1 PF-tree示例图

对FP-tree的频繁项集的挖掘过程就像是构建树的逆过程。从叶子节点出发，频繁项集就是前缀与叶子节点的组合。要注意的是，需要删除非频繁前缀。

## 4 宠物信息与数据流向