**PathFinder算法**：Path-Finder算法正是一种对复杂网络进行剪边压缩的网络优化算

法(还有聚类压缩等算法)。该算法最初由SchvaneveldtR.W.等于1988年提出,采用剪掉重要度较小的边的方法,抽取出复杂网络中实体间的主要关联关系。与其他剪边算法(如最少生成树算法等)相比,得到的网络图主要结构更加精确,在网络链接分析、检索界面设计、社会网络分析、认知结构表示、引文。网络分析、知识可视化等领域有着广泛的应用。

该算法由两个基本参数确定：**r**和**q**。**r**决定两个**不直接相连**的节点之间**距离测度**方法，**q**限定一个节点在在一个路径中可以经过的**最多节点数**。

**PathFinder算法如何过滤非重要节点**：主路径分析方法能够帮助读者快速把握一个学科领域的关键发展轨迹。主路径：从起点到终点之间累积最大遍历权重的路径。主路径上的节点应该是“重要节点”。二是主路径应该承载“最大信息量”。

**Spark平台**：Spark基于RDD实现了一体化、多元化的大数据处理体系，是目前最热门最高效的大数据领域的计算平台。Spark框架完美融合了SparkSQL、SparkStreaming、MLLIB、GraphX子框架，使得各子框架之间实现了数据共享和操作。

**Bayes统计**：贝叶斯是一种基于概率统计的学习算法。简单地说,贝叶斯理论可以概括为:为了预见未来,必须看看过去。随着统计决策理论、信息论和经验贝叶斯理论方法的创立与应用,贝叶斯方法很快显示出它的优点,成为一个活跃的研究方向。朴素贝叶斯一般分类原理就是利用类别的先验概率确定未知文本类别的后验概率。

**word2vec**： 深度学习word2vec是Google的一款将词表征为实数值向量的高效工具,其利用深度学习的思想,通过训练把对文本内容的处理简化为K维向量空间中的向量运算,而向量空间上的相似度可以用来表示文本语义上的相似度。在实际应用场景中,word2vec可以用于计算词的相似度、文档的相似度等,也可以用来对文档进行情感分析等。

深度学习word2vec中有CBOW和Skip-gram两种神经网络框架可用于训练并得到参数,参数迭代更新后得到的就是每个字的词向量。

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**词向量**是一种分布式的低维实数向量,其基本思想是通过训练将语料型中的词语映射到N维实数向量。

**可视化**：**数据可视化**来源于科学计算可视化，科学家们不仅需要通过图形图像分析计算机算出的数据，而且需要了解计算过程中数据的变换。科学可视化指的是空间数据场的可视化，而**信息可视化**是指非空间数据场的可视化，比如社会信息化与网络网络应用广泛推动产生的数据。**知识可视化**是在科学计算可视化数据可视化信息可视化基础上发展起来的新兴研究领域应用视觉表征手段促进群体知识的传播和创新。**知识可视化的实质**是将人们的个体知识(暂且称为认知知识制品)以图解的手段表示出来形成能够直接作用于人的感官的知识的外在表现形式(显}生的物理知识制品)从而促进知识的传播和创新。

**结构化数据**：**结构化数据**、**非结构化数据**以及**半结构化数据**是对存储形式的一种数据类型分析，有助于企业细分行业案例，帮助存储合作伙伴更好地解决应用实施方案。**结构化数据**，即行数据，存储在数据库里，可以用二维表结构来逻辑表达实现的数据；简单来说就是数据库。**半结构化数据**，包括邮件、HTML、报表、资源库等等，典型场景如邮件系统、WEB集群、教学资源库、数据挖掘系统、档案系统等等。**非结构化数据**，不方便用数据库二维逻辑表来表现的数据。**非结构化数据**，包括视频、音频、图片、图像、文档、文本等形式。

**命名实体**：命名实体是文本中基本的信息元素,是正确理解文本的基础。

**实体消歧**：命名实体的歧义是指一个实体的指称项可以对应多个实体概念。

**网络爬虫**：是按照一定规则,自动地抓取万维网信息的程序或脚本,为后续的数据操作提供数据集。

**垂直搜索引擎工具**：我们用的是贝叶斯统计推断。

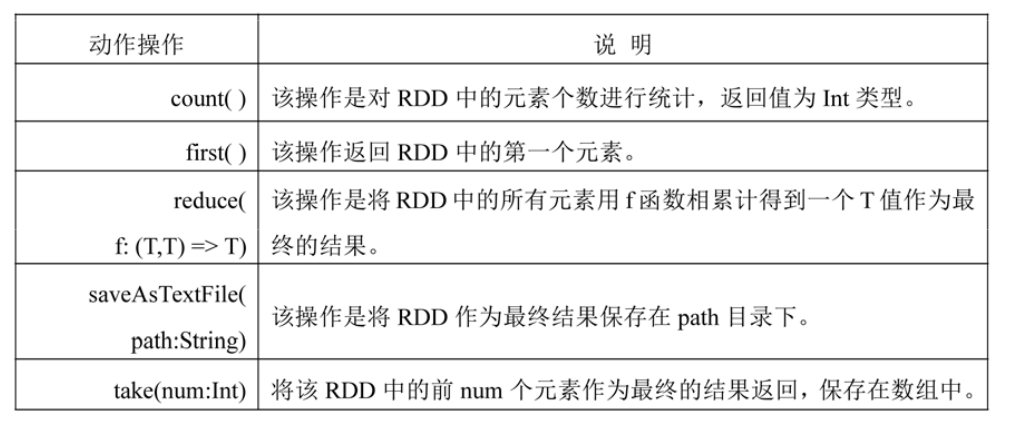
**通用知识图谱**：特点是不面向特定领域,可将其类比为“结构化的百科知识”。

**垂直知识图谱**：又称为行业知识图谱,特点是则面向特定领域,基于行业数据构建,强调知识的深度。

**Spark常用转换操作举例**：

****

**Spark常用动作操作举例**：

**RDD（弹性分布式数据集）**：该数据集具有容错功能，可以并行处理数据，它是一个高度抽象的数据结构，包含多个分区。（我记得那个慕课教程曾经举例，用在SparkSQL用RDD操作数据比在MySQL快许多？）

**word分词**：word分词是一个Java实现的分布式的中文分词组件，提供了多种基于词典的分词算法，并利用ngram模型来消除歧义。能准确识别英文、数字，以及日期、时间等数量词，能识别人名、地名、组织机构名等未登录词。能通过自定义配置文件来改变组件行为，能自定义用户词库、自动检测词库变化、支持大规模分布式环境，能灵活指定多种分词算法，能使用refine功能灵活控制分词结果，还能使用词频统计、词性标注、同义标注、反义标注、拼音标注等功能。提供了10种分词算法，还提供了10种文本相似度算法，同时还无缝和Lucene、Solr、ElasticSearch、Luke集成。

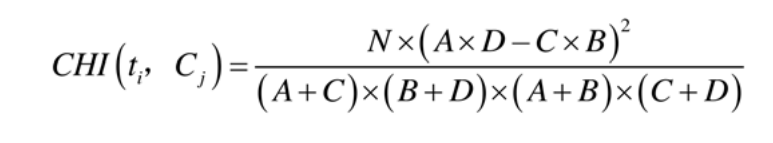
**ngram模型**：N-Gram（有时也称为N元模型）是自然语言处理中一个非常重要的概念，通常在NLP中，人们基于一定的语料库，可以利用N-Gram来预计或者评估一个句子是否合理。另外一方面，N-Gram的另外一个作用是用来评估两个字符串之间的差异程度。这是**模糊匹配**中常用的一种手段。（<https://blog.csdn.net/lengyuhong/article/details/6022053>）（<https://blog.csdn.net/baimafujinji/article/details/51281816>）

**特征降维方法**：所谓的降维就是，选出有代表性的特征词，这样就降低了维数。可以人工选择，也可以自动选择。

**χ2统计**：χ2 统计是用来度量两个变量的相关性的,值越大,表示两个变量的相关关系越紧密。

CHI衡量的是特征项ti和类别Cj之间的关联程度。特征对于某类的CHI统计值越高，则认为该特征对该类的类别表征程度就越强，反之，CHI值越小则认为该特征对该类的类别表征就越弱。

特征值对Cj的CHI值如下公式，该公式表示的是某词对于某个类的类别表征程度，最终的计算结果应该是一个矩阵。其中N表示训练语料中文本的总数，A表示属于某类且包含该词文档个数，B表示不属于某类但是包含该词文档个数，C表示属于某类但不包含某词文档个数，D表示既不属于某类也不包含某词的文档个数。



然后可以计算CHI值在每个训练类上的最大值。

**CHI方法的不足**

CHI是在文本分类中特征选择效果最好的方法之一，但是它也存在着不容

忽视的问题。

（1）从计算卡方值的过程就可以看出，经典CHI方法仅仅考虑了特征在训练集各类别中出现的文档频数，忽略了特征出现在文档中的词频，这样就可能提高某些对分类意义不大但文档频率高的低频词的卡方值，而减小某些在少量文档中多次出现、类别代表性强但文档频率低的特征对应的卡方值，会造成特征词选择不具有代表性。例如，在体育类相关的文档中，存在两个特征词“橄榄球”和“记者”，不难想象“记者”这个词在新闻中出现的文档频率应该很大，而“橄榄球”作为一个在国内并不热门的运动，出现在体育类文档中的次数应该相对较小，但是相比于“记者”，显然应该是“橄榄球”与体育类更加相关，更能代表这个类别的特征，但是根据统计计算出的结果却恰恰相反。

CHI不能准确保留在目标类别少量文档出现，但单篇文档中多次出现的特征词，这些词很有可能是某个类别特有的特征词，具有很强的类别代表性，对分类的意义重大，在特征选择过程中应该予以保留，但是通过传统χ2 统计方法计算的卡方值却相对较小，极有可能在特征选择时被过滤掉，所以在这一方面需要对CHI公式做出改进。

（2）CHI公式中分母上的值(AD-BC)2,由于公式所求是平方，因此存两种情况，都可能会由公式计算得到较大的卡方值。若AD一BC>0，代表特征亡和类别c是正相关，此时使得AD一BC值越大的特征，越能够代表文本类别；若AD一BC<0，这就说明[和c负相关，此时c的值比D大越多，越说明特征词出现携带的是文档不属于该类别的信息。正负相关两种情况下都有可能得到较大的卡方值。

实际上，在训练语料库中普遍出现的特征词，可能在指定类别中的文档频率很低甚至为0，说明该特征词不能很好的表示指定类别，在特征选择的时候应该过滤掉这样的特征词，但在利用传统公式计算该特征对指定类的卡方值的时候，得出的结果可能比较大，会被选择保留作为该类的特征项。

（3）特征选择希望选出满足以下条件的特征词：待选择的特征集中分布在指定类中，在其他类别中出现的可能性很小或者没有；另外在指定类别内部，大部分文档都包含该特征，而不是仅仅有少量包含该特征。经典CHI方法没有考虑到包含该特征项的文本在训练集中的分布，导致类间分布集中、类内均匀分布的特征并没有在计算公式中赋予较大的卡方值，从而影响了选择的特征子集的类别区分度，以此特征子集来训练分类器则会直接影响分类函数的构造，降低分类性能。要解决这个问题，需要对CHI做出适当改进。

**目前CHI方法改进方向**

针对经典CHI方法存在的不足，许多学者都对其做出了改进。通过研究分析现存的改进方法可以看出对CHI方法改进主要有两种不同的思路。

（1）结合CHI与其他特征选择方法

文本处理领域多种不同特征选择方法，它们各有优势和不足，改进CHI方法首先想到的就是与其他特征选择方法结合，取长补短来获得更好的效果。例如，PhayungMeesad等分析文档频率和信息增益方法的优点，并结合CHI方法提出一种组合方法，优化了CHI模型的性能：王光等将IG与CHI方法相结合，提出了一种综合特征选择算法CCIF，并在实验中证明其有效性：清华大学的李粤等通过分析CHI的不足，有针对性的利用互信息法调节计算结果，并将改进后的联合特征选择的算法应用在对网页的分类中，实验获得较好结果。

（2）优化CHI公式

另一种改进CHI的方法就是针对其存在的不足，在公式中增加一些参数来调整卡方值以达到更好的效果。目前国内相关研宄成果也很多，比如，张鹏招等利用频度、集中度、分散度三个指标对CHI进行优化，并通过实验结果证明该优化方法明显提高了查全率、查准率及FI的值；邱云飞等对特征在类内分布情况进行研究分析，提出了Var-CHI的特征选择方法并实验证明了其可行性；裴英博等考虑了特征与类别负相关的对分类性能的影响，提出了新的频度、集中度、分散度的计算公式，并且只考虑CHI与类别正相关的情况，最后通过实验证实改进方法的正确性和有效性。

关于χ2 统计，最好读读《大数据文本信息分类方法研究与实现\_邹游》和《基于VSM的文本分类系统的设计和实现\_孔振》这两篇论文，发的文档里面有。

**贝叶斯方法文本分类**：上面有。

**贝叶斯统计推断**：上面提了，注意看。

**SVM（支持向量机）**：在机器学习中，支持向量机（英语：support vector machine，常简称为SVM，又名支持向量网络）是在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。给定一组训练实例，每个训练实例被标记为属于两个类别中的一个或另一个，SVM训练算法创建一个将新的实例分配给两个类别之一的模型，使其成为非概率二元线性分类器。SVM模型是将实例表示为空间中的点，这样映射就使得单独类别的实例被尽可能宽的明显的间隔分开。然后，将新的实例映射到同一空间，并基于它们落在间隔的哪一侧来预测所属类别。

除了进行线性分类之外，SVM还可以使用所谓的核技巧有效地进行非线性分类，将其输入隐式映射到高维特征空间中。

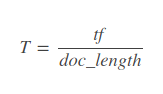
当数据未被标记时，不能进行监督式学习，需要用非监督式学习，它会尝试找出数据到簇的自然聚类，并将新数据映射到这些已形成的簇。将支持向量机改进的聚类算法被称为支持向量聚类，当数据未被标记或者仅一些数据被标记时，支持向量聚类经常在工业应用中用作分类步骤的预处理。（<https://blog.csdn.net/v_july_v/article/details/7624837>）

**VSM（向量空间模型）**：在该方法中，我们把一篇文章抽象成一个向量。假设向量由n个词组成，每个词的权重是kn。假设文章D，那么文章D=(k1,k2,k3,k4,k5.....kn),这是一个多维的向量。如果维数很多，将来计算起来很麻烦，我们需要**降维处理**。

**TF/IDF**：

**词频(tf，Term Frequency)**

一个词在一个句子中出现的次数越多，那么这个词在描述这个句子的含义方面贡献度越大，可通过下面两个式子中的一个来计算每个词的词权重：

****

**文档频率(df)**

文档频数(Document Frequency, DF)是最为简单的一种特征选择算法,它指的是在整个数据集中有多少个文本包含这个单词。在训练文本集中对每个特征计一算它的文档频次，并且根据预先设定的阑值去除那些文档频次特别低和特别高的特征。文档频次通过在训练文档数量中计算线性近似复杂度来衡量巨大的文档集，计算复杂度较低，能够适用于任何语料，因此是特征降维的常用方法。

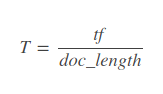
在训练文本集中对每个特征计算它的文档频数,若该项的DF 值小于某个阈值则将其删除,若其DF 值大于某个阈值也将其去掉。因为他们分别代表了“没有代表性”和“没有区分度”2 种极端的情况。DF 特征选取使稀有词要么不含有用信息,要么太少而不足以对分类产生影响,要么是噪音,所以可以删去。DF 的优点在于计算量很小,而在实际运用中却有很好的效果。缺点是稀有词可能在某一类文本中并不稀有,也可能包含着重要的判断信息,简单舍弃,可能影响分类器的精度。

文档频数最大的优势就是速度快,它的时间复杂度和文本数量成线性关系,所以非常适合于超大规模文本数据集的特征选择。不仅如此,文档频数还非常地高效,在有监督的特征选择应用中当删除90%单词的时候其性能与信息增益和x2 统计的性能还不相上下。DF 是最简单的特征项选取方法, 而且该方法的计算复杂度低, 能够胜任大规模的分类任务。

但如果某一稀有词条主要出现在某类训练集中,却能很好地反映类别的特征,而因低于某个设定的阈值而滤除掉,这样就会对分类精度有一定的影响。

逆文档频率(idf)

通常来说，如果一个词在越多的文档中出现过，那个这个词对某一个文档的贡献度应该就越小，也就是通过这个词来区分文档的区分度越小，可以用逆文档频率(idf)来度量这个概念。先定义另一个概念，文档频率df，表示包含某个词的文档的数目, N表示文档的总数量。逆文档频率计算公式如下：

****

**TF/IDF**

TF/IDF（term frequency/inverse document frequency) 的概念被公认为信息检索中最重要的发明。在搜索、文献分类和其他相关领域有广泛的应用。（<https://blog.csdn.net/quicmous/article/details/71263844>）

**RDD**：

### **优势**[[edit](https://www.semantic-mediawiki.org/w/index.php?title=Help:使用SPARQL和RDF存储&action=edit&section=2)]

在特定的维基站点之中是否要采用某种RDF存储，取决于许多的因素，包括所采用的具体的RDF数据库。不过，我们还是可以合理地期望获得下列优势：

* **查询性能更佳**  
   RDF存储在设计上旨在采用[SPARQL](https://www.semantic-mediawiki.org/wiki/SPARQL)查询语言来应答查询。与采用关系数据库的SQL查询语言相比，采用这种语言则可以更为自然地表达SMW查询。从这种意义上来说，SMW查询乃是RDF数据库系统最为典型的用例。而且，对于SMW查询来说，许多针对关系数据库查询的重要优化方法毫无用处，或者起着误导的作用。因此，可以期望，RDF存储应当可以提供极佳的查询性能。
* **额外的接口**  
   支持SPARQL标准的RDF存储还允许其他应用程序针对自己的数据提出SPARQL查询，且无须借助于SMW网络前端（web frontend）。这使得我们可以在其他应用程序当中有效地利用维基站点数据。有些具备SPARQL能力的数据库还进一步支持（部分的）[OWL 2](https://www.semantic-mediawiki.org/wiki/OWL)本体语言，并提供相应的接口来查询所存储的数据（比如，借助于[OWL链接协议（OWL Link protocol）](http://www.owllink.org/" \t "_blank)）。语义网应用程序也可以利用许多的公共编程库（common programming libraries）（如[librdf](http://librdf.org/" \t "_blank)或[OWL API](http://owlapi.sourceforge.net/" \t "_blank)），从而有助于在较低的层次上将它们与其他的工具集成起来。
* **推理功能以及基于本体的数据存取能力**  
   RDF Schema和 OWL之类的语义网语言，比如通过允许声明派生类（derived classes）或声明进一步的属性特性（property characteristics）（如属性的可递性<transitivity>），还为建模提供了额外的表达功能。有些具备SPARQL能力的数据库还能够*评价*这些查询应答功能；比如，对于基于本体的数据存取（ontology-based data access，OBDA）来说，利用语义建模构造（semantic modeling constructs），依据数据来创建“虚拟视图（virtual views）”的方法。
* **数据集成和本体重用**  
   可以在RDF数据库之中存储由SMW来更新的额外数据。这样，RDF存储就可以作为数据集成和本体重用的一种平台来发挥作用。
* **计算资源的物理隔离**  
   采用与MediaWiki之中不同的数据库后端，为在多个服务器之间分配任务提供了一种简便的方法。因而，尤其是可以防止复杂查询影响维基站点的基本操作，即使是它们意外地消耗了惊人数量的计算能力，也就是说，在它们搞垮了为RDF数据库提供主机托管服务的服务器的情况下。

### **缺点**[[edit](https://www.semantic-mediawiki.org/w/index.php?title=Help:使用SPARQL和RDF存储&action=edit&section=3)]

然而，同时也存在着许多可能的缺点：

* **存储要求更高**  
   只是在RDF数据库之中对数据进行了镜像，而不是从SQL数据库之中删除了数据。因而，需要额外的存储空间。
* **额外的维护工作**  
   RDF 后端在SMW之中的安装并不难，但要运行额外的数据库管理系统，仍要花费一些精力。
* **有关性能和稳定性的疑问**  
   目前，已有许多工业级的RDF数据库可用，其中有些还是免费或开源的。然而，将这些系统与SMW配合使用的经验还是有限的，因而在决定为大型SMW应用程序采用特殊的后端之前，进行一些测试还是会有所帮助。

幸运的是，无须花费太多精力，即可在基于SQL的存储后端与基于RDF的存储后端之间来回切换，因此，在试用一段时间之后，还可以再做定夺。

（<https://blog.csdn.net/u011801161/article/details/78833958>）

**监督学习和非监督学习的区别**

如果我们想要预测的是离散值，例如“好瓜”“坏瓜”，此类学习任务称为“**分类**”；如果想要预测的是连续值，例如西瓜成熟度0.95、0.37，此类学习任务称为“**回归**”。

学得模型后，使用其进行预测的过程称为“**测试**”，被预测的样本成为“**测试样**本”。例如在学得f后，对测试例x，可得到其预测标记y=f(x)。

我们还可以对西瓜做“**聚类**”，即将训练集中的西瓜分为若干组，每组称为一个“**簇**”；这些自动形成的簇可能对应一些潜在的概念划分，例如“浅色瓜”“深色瓜”，甚至“本地瓜”“外地瓜”。这样的学习过程有助于我们了解数据内在的规律，能为更深入地分析数据简历基础。需说明的是，在聚类学习中，“浅色瓜”“本地瓜”这样的概念我们事先是不知道的，而且学习过程中使用的训练样本通常不拥有标记信息。

根据训练数据是否拥有标记信息，学习任务可大致分为两大类：“**监督学习**”和“**非监督学习**”，分类和回归是前者的代表，而聚类是后者的代表。再解释一下，监督学习其实就是我们对输入样本经过模型训练后有明确的预期输出，非监督学习就是我们对输入样本经过模型训练后得到什么输出完全没有预期。结合西瓜的例子，监督学习就是我们知道经过模型训练后会分为好瓜或者坏瓜，而非监督学习则会将西瓜聚类为几种我们之前没有明确定义的瓜，如“浅色瓜”“外地瓜”。

**各种学习函**数：（<https://blog.csdn.net/tsyccnh/article/details/76769232>）

**各种优化函数**：（<https://blog.csdn.net/qq_21460525/article/details/70146665>）

**重点关注论文**：《基于Hadoop和SVM算法的中文文本分类的研究与实现\_张磊磊》、大数据文本信息分类方法研究与实现\_邹游》、《基于VSM的文本分类系统的设计和实现\_孔:振》以及所有参考文献。

**训练函数和学习函数区别**：训练函数和学习函数是两个不同的函数 ，网络设置中两个都有。简单的说，训练函数确定调整的大算法，是全局调整权值和阈值，考虑的是整体误差的最小； 学习函数决定调整量怎么确定，是局部调整权值和阈值，考虑的是单个神经元误差的最小。所以两者不冲突，可以一样也可以不同，就像你绕着楼跑步时，地球也在绕着太阳跑，是局部与整体的区别，既有联系又有区别，辩证统一。

训练函数是如何让误差最小的一些算法，如梯度下降，共轭梯度，这里强调算法。学习函数是指，权值和阈值的调整规则，或者称更新规则。训练函数求得权值或阈值之后，由学习函数进行调整，然后再由训练函数训练新的权值或阈值，然后再调整，反复下去。