The **Jaccard index**, also known as the **Jaccard similarity coefficient** (originally coined *coefficient de communauté* by [Paul Jaccard](http://en.wikipedia.org/wiki/Paul_Jaccard)), is a [statistic](http://en.wikipedia.org/wiki/Statistic) used for comparing the similarity and diversity of [sample](http://en.wikipedia.org/wiki/Sample_%28statistics%29) sets.

The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the [intersection](http://en.wikipedia.org/wiki/Intersection_%28set_theory%29) divided by the size of the [union](http://en.wikipedia.org/wiki/Union_%28set_theory%29) of the sample sets:

 J(A,B) = {{|A \cap B|}\over{|A \cup B|}}.

The [MinHash](http://en.wikipedia.org/wiki/MinHash) min-wise independent permutations [locality sensitive hashing](http://en.wikipedia.org/wiki/Locality_sensitive_hashing) scheme may be used to efficiently compute an accurate estimate of the Jaccard similarity coefficient of pairs of sets, where each set is represented by a constant-sized signature derived from the minimum values of a [hash function](http://en.wikipedia.org/wiki/Hash_function).

The **Jaccard distance**, which measures *dis*similarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union:

 J_{\delta}(A,B) = 1 - J(A,B) = { { |A \cup B| - |A \cap B| } \over |A \cup B| }.

例如有两个对象A,B，

A = 0101100110

B = 1010010010

可以看成A 和 B 有10 attributes，都是binary的，只有1,0，存在和不存在之分，这里A,B的形似度，可以这样计算：

M11 表示ＡＢ在相同属性都是1 的出现次数；

M10 表示ＡＢ在相同属性，A为1，B为0的出现次数；

M01 表示AB 在相同属性，A为0，B为1的出现次数；

M00 表示AB 在相同属性都是0的出现次数；

这里 显然有 M11 + M10 + M01 + M00 = n

Jaccard( A , B ) = ( M11 )/( M10 + M01 + M11 ) -🡪 为什么是M11，不包含M00呢？想象下

A={ a } ； B = { b } ， 如果n为4（abcdef），这样M00 = 4；M11=0；如果 M00 + M11计算， (4+0) / 6；AB挺相似，其实不然，AB集合不相似，因为我们从，AB都有的元素来看，而不是AB没有的元素来看相似性。

在数据挖掘中经常需要用到比较两个东西的相似度。比如搜索引擎要避免非常相似的文档出现在结果的前几页，再比如很多网站上都有的“查找与你口味相似的用户”、“你可能喜欢什么什么”之类的功能。后者其实是很大的一块叫做“协同过滤”的研究领域，留待以后详谈。

首先我们定义两个集合S,T的Jaccard相似度: Sim(S,T) = |S,T的交集| / |S,T的并集|。直观上就容易感觉出这是一个很简单而且比较合理的度量，我不清楚有没有什么理论上的分析，在此省略。下面先主要说一下文档的相似度。

如果是判断两个文档是否完全相同，问题就变得很简单，只要简单地逐字符比较即可。但是在很多情况下并不是这样，比如网站文章的转载，主体内容部分是 相同的，但是不同网页本身有自己的Logo、导航栏、版权声明等等，不能简单地直接逐字符比较。这里有一个叫做Shingling的方法，其实说起来很 圡，就是把每相邻的k个字符作为一个元素，这样整篇文档就变成了一个集合。比如文档是"banana"，若k=2，转化以后得到集合为 {"ba","an","na"}，于是又变成了前述集合相似度的问题。关于k值的设置，显然过小或过大都不合适，据说比较短的比如email之类可以设 k=5，比如长的文章如论文之类可以设k=9。

当然，这是一个看上去就很粗糙的算法，这里的相似度比较只是字符意义上的，如果想进行语义上的比较就不能这么简单了（我觉得肯定有一摞摞的 paper在研究这个）。不过同样可以想见的是，在实际中这个粗糙算法肯定表现得不坏，速度上更是远优于复杂的NLP方法。在实际工程中，必然糙快猛才是 王道。