# 句子相似度计算在FAQ中的应用

王洋　秦兵　郑实福

(哈尔滨工业大学321信箱，哈尔滨 150001)

E-mail: {wy,qinb,zsf}@ir.hit.edu.cn

**摘要**：本文设计并实现了一个基于常问问题库的中文问答系统。对用户以自然语言输入的问题，该系统能够自动地在FAQ（Frequently-Asked Question）库中寻找候选问题集，通过计算句子相似度，将匹配的答案返回给用户。该系统还能够自动地更新和维护FAQ库。文中着重介绍了用于查找候选问题集的数据结构以及句子相似度的计算方法。

**关键词**：自动问答；常问问题库；候选问题集；句子相似度

## 引言

自动问答系统是目前自然语言处理领域一个非常热的问题，它即能够让用户用自然语言句子提问，又能够为用户返回一个简洁、准确的答案，而不是一些相关的网页。因此，自动问答系统和传统的依靠关键字匹配的搜索引擎相比，能够更好地满足用户的检索需求，更准确地找出用户所需的答案，具有方便、快捷、高效等特点。在国际上每年一度的文本信息检索(TREC)会议上，自动问答(Question Answering Track)是最受关注的主题之一。

常问问题库 (FAQ)是很多自动问答系统中的一个组成部分。它把用户常问的问题和相关答案保存起来。这样，对于用户输入的问题，可以首先在常问问题库中查找答案。如果能够找到相应的问题，就可以直接将问题所对应的答案返回给用户，而不需要经过问题理解、信息检索、答案抽取等许多复杂的处理过程。本文将对自动问答系统中FAQ的设计和实现方法做一全面介绍，并着重介绍了其中的句子相似度计算。本文所介绍的句子相似度的计算方法不仅能够用于FAQ的检索，还能够用于自动问答的其它阶段，本文简要地介绍了其在答案查找中的应用。

## 1 系统概述

系统主要包含三个部分：候选问题集的查找，句子相似度计算，FAQ库的更新。

## 2 候选问题集的查找

这一步骤的目的是要从常问问题库(FAQ)中找出若干个候选的问题组成候选问题集，以缩小查找的范围，使后续的相似度计算等较复杂的处理过程都在候选问题集这个相对较小的范围内进行。在本系统中，我们选出FAQ中50%的问句作为候选问题集。设用户输入的问句（简称为目标问句）中共有 $n$ 个词：$W_1$、$W_2$、…、$W_n$。FAQ库中共有 $m$ 个问句，第 $i(1\le i\le m)$ 个问句含有 $n_i$ 个词：$Q_1$、$Q_2$、…、

$Q_{n_i}$。第 $i$ 个问句和目标问句之间重叠的词个数记为 $Num_i$，即 $Num_i = |\{W_1, W_2, ..., W_n\} \cap \{Q_1, Q_2, ..., Q_n\}|$。我们将 $Num_i$ 值最大的前 50%的 FAQ 问句选出来，组成候选问题集。

计算 $Num_i$ 时，如果将 FAQ 库中的问句一一读出来和目标问句进行比较，效率是比较低的。对于目标问句中的某个词，为了能够快速地统计 FAQ 库中究竟有多少问句含有这个词，我们设计了如图 1 所示的数据结构：

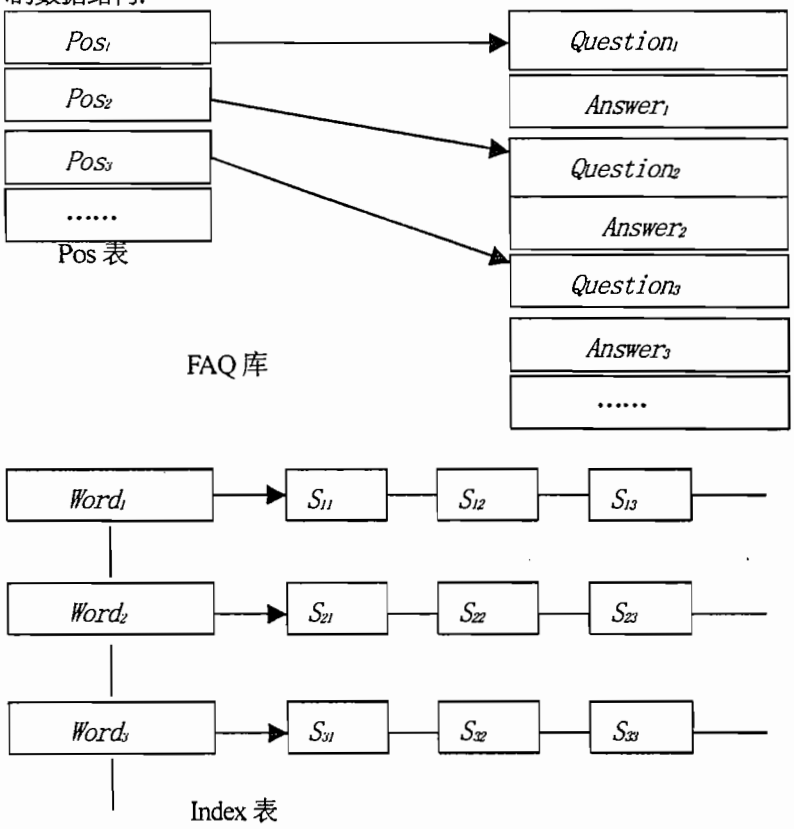


图 1 用于查找候选问题集的数据结构

图 1 中的 FAQ 库记录了所有原始的问题、答案对，Pos 表记录了 FAQ 库中每个问句在库文件中的位置。Index 表中的 $Word_1$、$Word_2$ ……是 FAQ 库中的问句所包含的词经过排序后所形成的链表。而每个 $Word_i$ 指向一个 S 链表，这个 S 链表中的每个节点记录 FAQ 库中含有 $Word_i$ 的一个问句的句子号。

在实际的检索过程中，对于目标句子中的一个词 W，我们首先寻找它在 Word 链表中的位置。由于 Word 链表是有序的，我们可以很容易地利用折半查找等方法在 O(logn)的时间复杂度内找到目标。不妨设找到的节点为 $Word_k$，沿着 $Word_k$ 所指向的 S 链表，我们就可以统计出有哪些 FAQ 库中的问句包

含$Word_k$。对目标问句中的每一个词都进行这样的处理之后，我们就可以进一步计算出上面提到的$Num_i$的值。

接下来，我们找出Num值最大的50%个问句的句子号，通过Pos表，可以在很容易地将FAQ库的文件指针移到相应的问句处，并把问句读出。

## 3 句子相似度计算

候选问题集确定后，下一步是要从这个集合中找出和用户输入的问句（这里称为目标问句）最相似的问句。我们所用的方法是计算候选问题集中每个问句和目标问句之间的相似度，对应的相似度最大的问句就是我们要找的句子。

计算相似度的方法有很多，这里我们综合运用了两种方法来计算句子的相似度，一种是基于向量空间模型的TFIDF方法，另一种是基于语义的方法。下面具体介绍这两种方法。

### 3.1 基于向量空间模型的TFIDF方法

在信息检索领域中，基于向量空间模型的TFIDF方法被广泛地用来计算文本之间的相似度。若FAQ中所有问句包含的所有的词为$w_1$、$w_2$、…、$w_n$，则FAQ中的每一个问句都可以用一个n维的向量$T=<T_1,T_2,...,T_n>$来表示。其中，$T_i(1\le i\le n)$的计算方法为：设n为$w_i$在这个问句中出现的个数，m为FAQ中含有$w_i$的问句的个数，M为FAQ中问句的总数，那么$T_i=n\times\log(M/m)$。从这个式子中可以看出，出现次数多的词将被赋予较高的n值，但这样的词并不一定具有较高的$\log(M/m)$值。例如，在汉语中“的”出现的频率非常高，即tf值(n值)很大，但由于“的”在很多文档中都出现，它对于我们分辨各个文档并没有太大的帮助，它的idf值$(\log(M/m))$将是一个很小的数。因此，这种方法综合地考虑了一个词的出现频率和这个词对不同文档的分辨能力。

用同样的方法，我们可以计算目标问句的n维向量$T'=<T_1',T_2',...,T_n'>$。得到$T$和$T'$后，它们所对应的两个句子之间的相似度就可以利用$T$和$T'$这两个向量之间夹角的余弦值来表示。

$$Similarity(T,T')=\frac{\sum_{i=1}^{n}T_i\times T_i'}{\sqrt{\sum_{i=1}^{n}T_i^2\sum_{i=1}^{n}T_i'^2}} \tag{1}$$

TFIDF方法综合考虑了不同的词在问句中的出现频率(tf值)和这个词在整个FAQ库中对不同句子的分辨能力(idf值)。这种方法不需要任何对文本内容的深层理解，它能够在我们的FAQ中应用，很重要的一个原因是我们的FAQ库是非受限域的自然语言文本，而且FAQ库通常都很大。

## 3.2 基于语义的相似度计算方法

TFIDF 是信息检索领域常用的方法，并且一般来说能够产生较好的效果。但在我们的这个系统中，单靠 TFIDF 的方法还不能达到我们所预期的结果。原因有两个：第一，TFIDF 方法只有当句子所包含的词比较多时效果才好。因为 TFIDF 是一种统计的方法，只有当句子包含的词数越多，相关的词才会重复出现，这种统计方法的效果才能体现出来。而在我们的 FAQ 中，我们所面对的是单个的问句，问句中包含的词的个数往往不足以体现这种方法的效果；第二，TFIDF 方法只考虑了词在上下文中的统计特性，而没有考虑词本身的语义信息。例如，“西红柿是什么颜色？”和“番茄是什么颜色？”所表达的应该是完全相同的意思，因为“西红柿”和“番茄”在语义上是等价的。由于 TFIDF 没有考虑到这种语义信息，因此具有一定的局限性。这将在本文 3.3.4 的实验结果中体现出来。

基于上述两点，我们又引入了基于语义的相似度计算方法。

### 3.2.1 语义知识资源

计算语义相似度，需要一定的语义知识资源作为基础。在英语中，人们通常用 WordNet。这里我们用董振东和董强先生创建的知网(HowNet)作为系统的语义知识资源。知网是一个以汉语和英语所代表的概念为描述对象，以揭示概念与概念之间以及概念所具有的属性之间的关系为基本内容的常识知识库。它是一个网状的有机的知识系统。

语义词典是知网的基础文件，在这个文件中每一个词语的概念及其描述形成一个记录。每一个记录都包含词语、词语例子、词语词性、概念定义等四项。我们这里主要用到的是概念定义(即 DEF)这一项。

计算句子之间的语义相似度，首先要确定句子中的词在这个句子中所表达的语义。例如，“打毛衣”中的“打”作为“编织”的意思，而“打酱油”中的“打”作为“买”的意思，而我们需要确定“打”这个词在不同的句子中的不同含义。再比如，“西红柿”和“番茄”是两个不同的词，但在语义的层面上，这两个词是相同的。这一步的工作称为语义消歧。我们用的是哈尔滨工业大学计算机学院信息检索实验室所做语义消歧系统。该系统能够对经过分词和词性标注的句子进行语义消歧，并在每个词后面标注上相应的语义号。例如：

对于句子：

哈尔滨/nd 在/p 什么/r 地方/ng ？/wj

经过语义消歧后变为：

哈尔滨/17 在/1269 什么/468 地方/17 ？/-1

每个语义号都对应知网中的一个义原。例如，17 对应的义原为“place| 地方”，1269 对应的义原为“{location}”，468 对应的义原为“aValue|属性值,kind|类型”，-1 表示在知网中找不到这个词（例如“公转”）或者这个词没有有价值的语义信息(例如标点符号)。

对于上面所说的“打酱油”中的“打”，语义号为 348(buy|买)，“打毛衣”中的“打”语义号为 525(weave|辫编)。由此可以看出，语义消歧能够挖掘出一个词在上下文中确切的含义，而不是仅仅停留在词的表面。这位后面的工作奠定了基础。

除了语义词典，知网中还提供了义原分类树，义原分类树把各个义原及它们之间的联系以树的形式组织在一起，树中父节点和子节点的义原具有上下位的关系。我们可以利用义原分类树计算两个词之间的语义距离。知网中存在 Entity、Event、Attribute 等 11 棵义原树。但有些义原树，例如 Converse、Antonym 等，里面的义原没有父子关系，并不体现上述的词与词之间的上下位特征，因此无法使用。我们在 11 棵义原树中总共选取了以下 6 棵义原树用来计算词的语义距离：Entity、Event、Attribute、Attribute Value、Quantity、Quantity Value。

### 3.2.2 词与词之间语义相似度的计算

首先需要计算两个词之间的语义距离。这里，我们把语义距离定义为两个词对应的义原在义原树中的最短距离。如果两个词中有一个词的义原无法在 3.2.1 中提到的六棵义原树中找到，或者两个词的义原

分别处于两个不同的义原树，我们认为这两个词之间的语义距离为∞。

设两个词 U、V 之间的语义距离为 p，那么 U、V 之间的相似度可以用公式(2)来计算：

$$s(U,V)=\begin{cases}H-(p\times(H-L)/D) & (p\neq\infty)\\ 0 & (p=\infty)\end{cases} \tag{2}$$

这里的 H 和 L 是两个词之间相似度可能取得的最大和最小值。在本系统中，令 H=1，L=0。D 是 U、V 所在的义原树的中两个义原的语义距离可能的最大取值。即如果某个义原树中深度最大的两个义原的深度分别为 D1、D2，那么这棵语义树的 D=D1+D2。注意，根据上面所说，当 p≠∞时，U、V 的义原必定是在同一棵义原树中，因此，关于 D 的定义是合理的。

**3.3.3 句子之间语义相似度的计算**

有了词与词之间的语义相似度，我们就可以来计算句子间的语义相似度。设两个句子 A 和 B，设 A 包含的词为 $A_1$、$A_2$、…、$A_m$，B 包含的词为 $B_1$、$B_2$、…、$B_n$。词 $A_i(1\le i\le m)$ 和 $B_j(1\le j\le n)$ 之间的相似度用 $s(A_i,B_j)$ 来表示，这样我们得到一个 $m\times n$ 的矩阵：

$$M(A,B)=\begin{bmatrix}s(A_1,B_1),s(A_1,B_2),\ldots,s(A_1,B_n)\\ \ldots\ldots\\ s(A_m,B_1),s(A_m,B_2),\ldots,s(A_m,B_n)\end{bmatrix}$$

利用这个矩阵，我们可以用公式(3)得到 A,B 两个句子之间的语义相似度 $s(A,B)$：

$$s(A,B)=\frac{\sum_{i=1}^{m}\max(s(A_i,B_1),s(A_i,B_2),\ldots,s(A_i,B_n))}{m} \tag{3}$$

最后，我们利用 TFIDF 算出的相似度和用语义算出的相似度的加权平均，就可以计算出两个句子最终的相似度。设利用 TFIDF 算出的句子相似度值为 t，利用语义算出的句子相似度值为 s，两个句子最终的相似度 m 可以表示为公式(4)：

$$m=\frac{tT+sS}{T+S} \tag{4}$$

T 和 S 是分别赋予 t、s 的权重。

**3.3.4 试验结果**

语义相似度的效果可以用下面的实验来说明。

FAQ 库中有两个句子：

S1=楼房如何建造?

S2=高尔夫球怎么打?

我们分别来看这二个句子和用户输入的一个句子“房子怎么盖？”(简称为 S)的相似度。

表 1　两种相似度计算结果的比较

| | TFIDF 计算的相似度 | 语义相似度 |
|---|---|---|
| S1 | 0 | 1 |
| S2 | 0.129466 | 0.0675436 |

我们可以看出，由于S1和S中没有任何相同的词，所以利用TFIDF计算出的相似度为0。但是S1和S表达的是同一个意思，这就不符合我们的要求了。而且利用TFIDF算出的值，S2和S的相似度大于S1和S的相似度，这显然是荒谬的。而利用语义得到S1和S的相似度为1，说明S1和S两个句子在语义上相同，而且S1和S的相似度远远大于S2和S的相似度，这完全符合实际的情况。从这个实验中，我们可以看出语义相似度在处理两个句子中相同的词很少，但两个句子中词的语义是相同的这种情况时，比TFIDF方法优越。

上述计算相似度的方法还用于问答系统中的其它阶段。例如，在答案查找过程中，我们可以通过计算问句和备选答案之间的相似度。例如，问题是“西红柿是什么颜色？”，如果有一个句子为“番茄是红色的。”，这两个句子通过上述的相似度计算方法就可以匹配上。而如果单纯地依靠关键词，“西红柿”和“番茄”是不可能匹配上的。

## 4 FAQ库的更新

利用3中介绍的方法计算出用户所输入的目标问句和候选问题集中每个问句的相似度，如果所有这些计算出来的相似度的最大值大于一定的阀值M，那么我们就认为最大的相似度所对应的问句和用户的目标问句问的是同一个问题。我们可以直接将这个问句对应的答案输出给用户。

如果最大相似度的值小于阀值M，我们就认为FAQ库中没有用户所问的问题，那么我们必须利用其他的方法（如信息检索，答案抽取等）来找出答案。如果能够找到答案，就可以将用户所问的这个问题和对应的答案加入FAQ库。

参考文献:

[1]Burke, R., Hammond, I., et al., Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System, Univ. of Chicago, Dept. of Computer Science Technical Report TR-97-05, 1997.

[2]Eugene Agichtein, Steve Lawrence, and Luis Gravano. Learning Search Engine Specific Query Transformations for Question Answering. In the Proceedings of the 10th World Wide Web Conference(WWW2001), 2001

[3]董振东，董强. 知网. http://www.keenage.com

[4]张刚，刘挺，郑实福，车万翔，秦兵，李生. 开放域中文问答系统的研究与实现. 中国中文信息学会二十周年学术会议. 2001

致谢 哈尔滨工业大学信息检索实验室的老师和同学对本文的完成提出了很多有益的建议，在此一并表示感谢.

作者简介： 王洋（1979－），男，河南郑州人，哈尔滨工业大学四年级学生。秦兵(1968－)，女，陕西华阴人，博士生，主要研究领域为信息检索，自动问答，多文档文摘。

# Sentence Similarity for Frequently-Asked Question

WANG Yang QIN Bing ZHENG Shifu

*(Box321, Harbin Institute of Technology, Harbin 150001,China )*

E-mail: {wy,qinb,zsf}@ir.hit.edu.cn

**Abstract:** In this paper, we describe the design and implementation of a question answer system based on FAQ (Frequently-asked Question). This system involves automatically searching for candidate question set, computing sentence similarity and returning the answer to the user. This system can also automatically update and maintain FAQ. We introduce the data structure used to search for candidate question set and the method to compute sentence similarity in detail.

**Key words:** question answering; FAQ; candidate question set; sentence similarity