东南大學 SOUTHEAST UNIVERSITY 网络空间安全学院 School Of Cyber Science and Engineering

# 东南大学模式识别课程作业

# 运用文本相似度实现主观题智能评阅

姓名：程晓宇　　学号：239762

# 1 前言

近年来，“AI+”带来的发展红利普惠民生，教育领域对于解放人工劳动力的智能阅卷需求亦愈发明显；尤其是职业教育越来越鼓励学生对知识的综合应用，而非单纯的知识水平考试，方案设计等主观题的考察占比越来越高。然而传统的阅卷软件，仅支持客观题的自动评分，对于主观题仍是通过各种软硬件手段将考生答案以文本、音像等形式呈现给阅卷教师，教师再据此打分。显然，这种阅卷方式在本质上仍属于人工阅卷。这类阅卷方式存在着阅卷的效率低、成本高，人工对主观题评阅有一定的主观性和不确定性等显著的问题。如今的教育行业对主观题智能评阅系统的需求非常明显，智能化的主观题评阅系统能够产生卓越的社会、经济、文化等多方位效益。

目前，国内的中文语义分析技术较多，但大部分都是近几年才逐渐出现的，都不能算是特别成熟。通过调查不难得知，国内主流的大数据语义分析平台中，较为出色的代表有北京理工大学的 NLPIR 大数据语义智能分析平台 [1]（原 ICTCLAS）和哈工大语言云（LTP）[2] 平台等。其中 NLPIR 平台针对大数据内容采编挖搜的综合需求，融合了网络精准采集、自然语言理解、文本挖掘和语义搜索的最新研究成果，拥有较全面的功能。虽然针对中文长文本语义分析的解决方案在不断提出和实施，但由于汉语的博大精深，这方面的探索和研究尚未进入深水区，许多情境，特别是具体场景的应用仍有较大开发空间。作为中文长文本语义相似度算法的具体应用，主观题评阅系统前景比较广阔，目前亟须解决的问题还有许多，举例如下：

（1）对于在不同语境中的相同词语不能做出有效判别。中文中一词多义，一义多词的现象普遍且繁复，同一个词在不同语境下表达的语义不一样，带有的情感色彩也会有差别，整个语句的含义也会因此天差地别。

（2）不带情感的停用词无法有效识别。例如：打开天窗说亮话。这本是不带有感情色彩和实际意义的话，如果单纯靠计算机识别，无法做到与人的判断相同的结果。

（3）新词新义的更新速度加快。互联网的快速发展，许多网络词语应运而生，并且不少网络词语已经成为官方用语，这使得计算机对语言的识别也必须跟上时代的步伐。

因此，本文通过借助问题和关键词更加多角度全方位地去辅助剖析主观题，设计一款智能化的主观题评阅系统，提供一个可推广的长文本语义相似度计算方案。该系统能够分析句子结构，识别句子中的否定词，并成功处理动词、形容词、副词等的双重否定或反义词与原词的相似性匹配。此外，它还能够处理常见解释性语句，实现词语与其解释之间的匹配。在评分方面，该系统实现了根据回答中命中关键词并符合答案意义的方式进行分数分配。总的来说，本研究超越了传统方法。通过提供一个全面的长文本语义相似度计算解决方案，包括对否定和解释性语句等语言细节的处理，所提出的系统不仅通用性强，而且适用于广泛的题型。这对于提高主观题评价的准确性和效率，并最终推动自动评分系统领域的发展具有重要意义。本研究的潜在应用范围涵盖了各种教育场景，为教育工作者提供了一个有价值的工具，以更为复杂和深刻的方式评估学生答案并提供反馈。

# 2 系统总体设计

系统整体分为算法模块和 web 服务端模块，在算法模块中，首先对于用户输入的问题，标准答案，关键词和学生回答，系统会在语料库和词典的辅助下通过 jieba，jiagu，synonyms 库以及基于 TextRank [3] 算法的关键词提取模块进行分词,提取关键词，关键词拆分；然后基于搜索 query 分析的同义词挖掘模块对于双重否定、同义词，反义词等语义进行识别和处理；最后结合基于改进 FastText [4] 模型计算相似度模块和不同关键词，如单词、多词，对应的打分机制计算出相应的得分。web 服务端则由前端的 vue [5] 框架和后端的 spring boot 框架以及 flask 框架搭建而成，数据库选用了 mysql 数据库通过 mybatis 和 springboot 后端整合，以上两个模块组成完整的应用平台。系统的总体框图如下图所示。

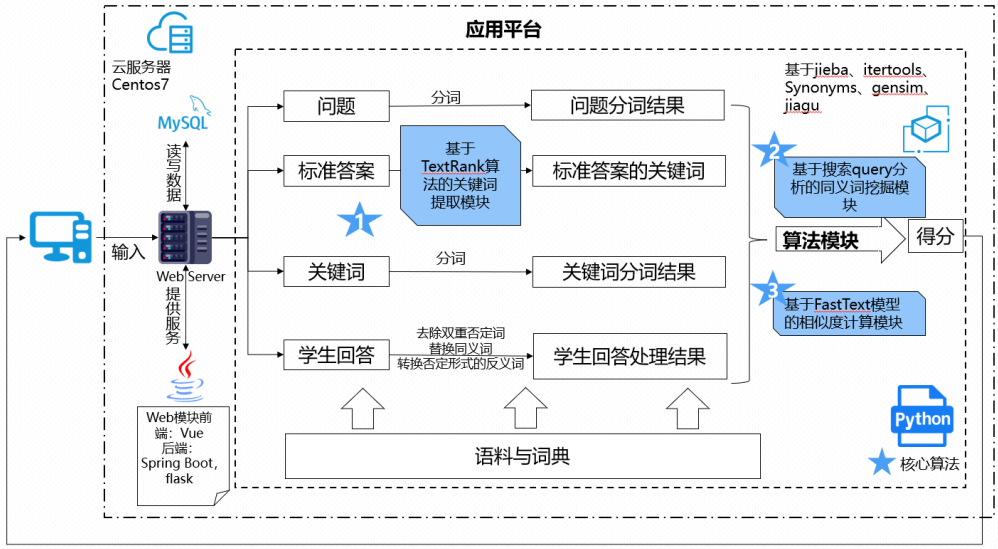



图 2.1 系统总体框图

## 2.1 算法（模型）技术路线

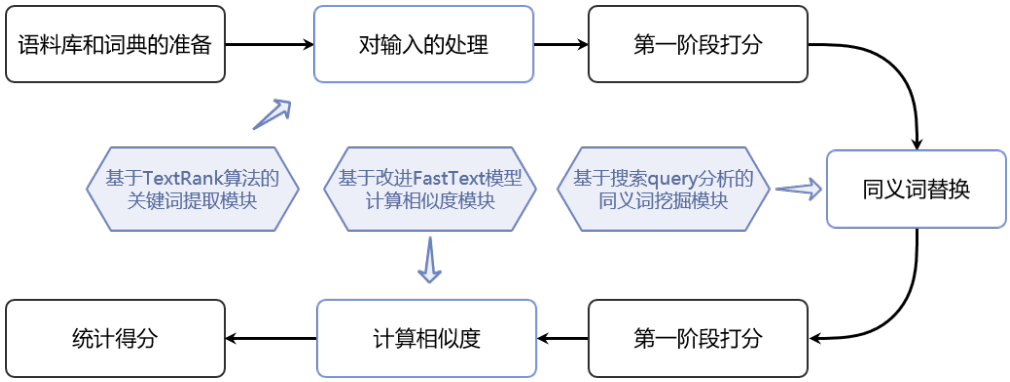



图 2.2 技术路线框架

算法程序主要可分为六个步骤：语料库和词典的准备、对输入的处理、第一阶段打分、同义词替换、第二阶段打分和计算相似度。

系统的输入包括问题、标准答案、关键词和学生回答。通过问题的分词结果过滤关键词和学生答案中某些对打分影响不大的词语。对于关键词，采用改进后的 TextRank [3] 关键词提取算法得到了标准答案中重要程度更高的关键词汇，并将关键词分词后得到了新的关键词列表（参照图 2.2）、关键词权重和语义记号。为了便于接下来的打分操作，去除学生答案中的双重否定词，替换同义词和带否定的反义词。

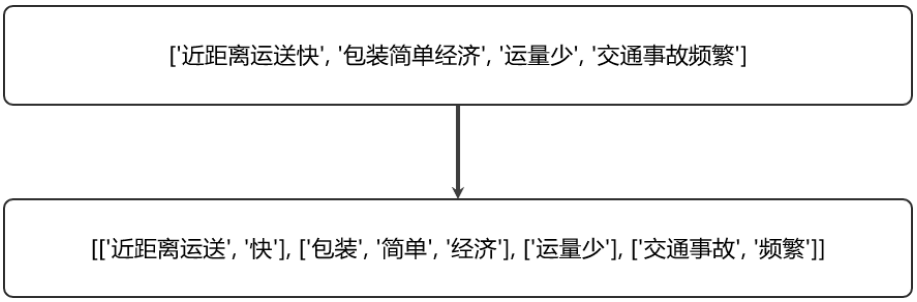



图 2.3 关键词的处理样例

第一阶段打分分为五个步骤，其中单个词是指处理过后的关键词某项只含一个词，两个词同理（参照图 2.3）。精确模式比非精确模式更为严格。在非精确模式下，针对单个词和两个词有不同的参数设定，以模拟真实情况。之所以将单个词和两个词的打分分开，是因为两者同时进行时，容易出现词语有交集或者在非精确模式下词语被误选等情况，经过试验后发现，将两者分开能很好地减少问题的出现。另外，在学生回答中的词语得分后会被替换成‘，’。最后一步的单个词拆分后精确模式的设计，是为了减少受分词的影响导致原本连在一起的词语被拆分导致不给分的情况出现。

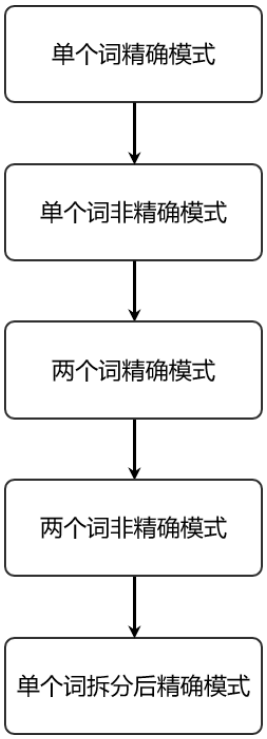



图 2.4 第一阶段打分流程图

在完成第一阶段打分后，由于学生回答含有同义词，两个词的打分效果不佳，此时系统基于搜索 query 分析的同义词挖掘模块 [6] 进行同义词替换，但是并非所有的词语都有寻找同义词的价值，为模拟真实场景只对动词、动名词和形容词寻找同义词。寻找方式为遍历左边 list 中的词语，寻找该词的同义词是否出现在右边 list 中，故存在两种寻找方向，如图 2.5 所示。尽管“出发点”不同，但是目的都是找出关键词和学生回答中的同义词对并替换，为了第二阶段打分做准备工作。

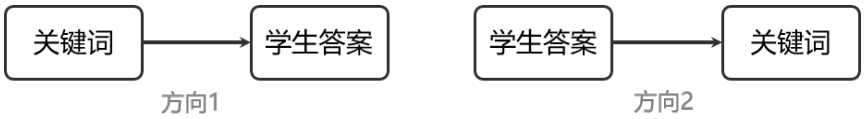



**图 2.5 同义词替换寻找方向**

在替换同义词后，再次对两个词的关键词进行精确和非精确模式的打分。三个词及以上的关键词一般不含专业词汇，多为一些短语搭配，容易出现学生使用同义词回答的情况，故将该部分的打分安排在此，并采用非精确模式打分。为了叙述的方便，后文中将三个词及以上称为多个词。

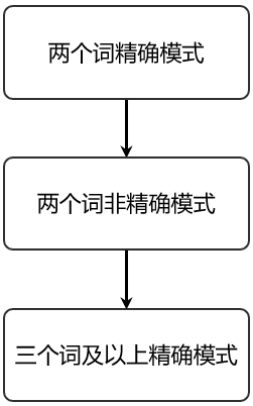



**图 2.6 第二阶段打分流程图**

如图 2.7 所示，经过两个阶段打分后的学生回答 list 剔除标点以及“回答”后与还未得分的关键词 list 通过“计算相似度方法”得到二者相似度，若相似度大于某个值且语意相同，则得到一个分数作为未得分关键词部分的分数补充。

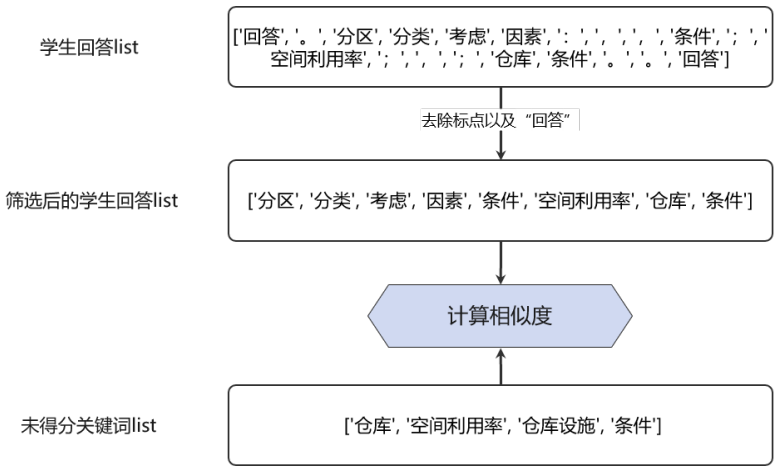



图 2.7 第二阶段打分流程图

# 3 算法详细设计

## 3.1 基于改进 FastText 模型计算相似度模块

### 3.1.1 模型介绍

#### 3.1.1.1 Softmax 回归

Softmax 回归 [7] 又被称作多项逻辑回归，它是逻辑回归在处理多类别任务上的推广。

在逻辑回归中，有 m 个被标注的样本: $\{(x^{(1)},y^{(1)}),\dots,(x^{(m)},y^{(m)})\}$，其中

$$h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$$

代价函数如下：

$$J(\theta)=-\left[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log\left(1-h_\theta(x^{(i)})\right)\right]$$

在 Softmax 回归中，类标是大于 2 的，因此在训练集 $\{(x^{(1)},y^{(1)}),\dots,(x^{(m)},y^{(m)})\}$，$y^{(i)}\in\{0,1,\dots,K\}$中，给定一个测试输入 x，假设应该输出一个 K 维的向量，向量内每个元素的值表示 x 属于当前类别的概率。具体地，假设 $h_\theta(x)$ 形式如下：

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=k|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} e^{\theta(j)^T x}} \begin{bmatrix} e^{\theta(1)^T x} \\ e^{\theta(2)^T x} \\ \vdots \\ e^{\theta(K)^T x} \end{bmatrix}$$

代价函数如下：

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=i}^{K} 1\{y^{(i)}=k\} \log \frac{e^{\theta(k)^T x^{(i)}}}{\sum_{j=1}^{K} e^{\theta(k)^T x^{(i)}}}\right]$$

其中 $1\{\cdot\}$是指示函数，即$1\{true\}=1$，$1\{false\}=0$。逻辑回归是 Softmax 回归在 K=2 时的特例。

#### 3.1.1.2 分层 Softmax

在标准的 Softmax 回归中，要计算 y=j 时的 Softmax 概率：$P(y=j)$，需要对所有的 K 个概率做归一化，这在|y|很大时非常耗时。

分层 Softmax 的基本思想是使用树的层级结构替代扁平化的标准 Softmax，使得在计算时，只需计算一条路径上的所有节点的概率值，无需在意其它的节点。

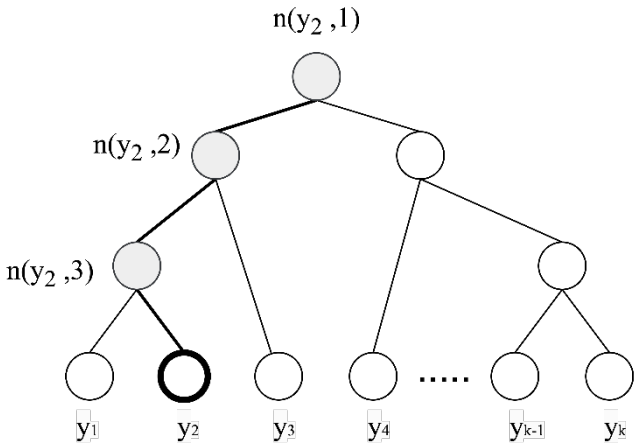



图 3.1 分层 Softmax 示意图

树的结构是根据类标的频数构造的霍夫曼树。K 个不同的类标组成所有的叶子节点，K-1 个内部节点作为内部参数，从根节点到某个叶子节点经过的节点和边形成一条路径，路径长度被表示为$L(y_i)$。于是，$P(y_i)$就可以被写成：

$$p(y_i) = \prod_{l=1}^{L(y_j)-1} \sigma([\![n(y_j,l+1) = LC(n(y_j,l))]\!] \cdot \theta_{n(y_j,l)}{}^T X)$$

其中：$\sigma(\cdot)$表示 sigmoid 函数，$LC(n)$表示 n 节点的左孩子；

$[\![x]\!]$是一个特殊的函数，被定义为：

$$[\![x]\!] = \begin{cases} 1 \text{ if} x == \text{true} \\ -1 \text{ if} x == \text{false} \end{cases}$$

$\theta_{n(y_j,l)}$是中间节点$n(y_j, l)$的参数；X 是 Softmax 层的输入。

图 3.1 中，高亮的节点和边是从根节点到$y_2$的路径，路径长度$L(y_2) = 4, P(y_2)$可以被表示为：

$$P(y_2) = P(n(y_2, 1), left) \cdot P(n(y_2, 2), left) \cdot P(n(y_2, 3), right)$$

$$= \sigma(\theta_{n(y_2,1)}{}^{T} X) \cdot \sigma(\theta_{n(y_2,2)}{}^{T} X) \cdot \sigma(-\theta_{n(y_2,3)}{}^{T} X)$$

于是，从根节点走到叶子节点，实际上是在做了 3 次二分类的逻辑回归。

通过分层的 Softmax，计算复杂度一下从|K|降低到$log|K|$。

#### 3.1.1.3 n-gram 特征

n-gram [8] 是一种基于语言模型的算法，基本思想是将文本内容按照字节顺序进行大小为 N 的滑动窗口操作，最终形成长度为 N 的字节片段序列。例如：

“今天你敲代码了吗”

相应的 bigram 特征为：

“今天” “天你” “你敲” “敲代” “代码” “码了” “了吗”

相应的 trigram 特征为：

“今天你” “天你敲” “你敲代” “敲代码” “码了吗”

注意一点：n-gram 中的 gram 根据粒度不同，有不同的含义。它可以是字粒度，也可以是词粒度的。上面所举的例子属于字粒度的 n-gram，词粒度的 n-gram 看下面例子：

今天 你 敲 代码 了 吗

相应的 bigram 特征为：

今天/你 你/敲 敲/代码 代码/了 了/吗

相应的 trigram 特征为：

今天/你/敲 你/敲/代码 敲/代码/了 代码/了/吗

#### 3.1.1.4 FastText 模型

**1．字符级别的 n-gram**

word2vec 把语料库中的每个单词当成原子的，它会为每个单词生成一个向量。这忽略了单词内部的形态特征，比如：“苹果醋” 和“苹果”，“apples”和“apple”，这两个例子中，两个单词都有较多公共字符，即它们的内部形态类似，但是在传统的 word2vec 中，这种单词内部形态信息因为它们被转换成不同的 id 丢失了。

为了克服这个问题，FastText 使用了字符级别的 n-grams 来表示一个单词。对于单词“apple”，假设 n 的取值为 3，则它的 trigram 有:

“<ap”， “app”， “ppl”， “ple”, “le>”

其中，<表示前缀，>表示后缀。于是，可以用这些 trigram 来表示“apple”这个单词，进一步，可以用这 5 个 trigram 的向量叠加来表示“apple”的词向量。

这带来两点好处：

（1）对于低频词生成的词向量效果会更好。因为它们的 n-gram 可以和其它词共享。

（2）对于训练词库之外的单词，仍然可以构建它们的词向量。可以叠加它们的字符级 n-gram 向量。

**2．模型架构**

FastText [4] 模型架构和 word2vec 的 CBOW 模型 [9] 架构非常相似，分为输入层、隐含层和输出层，如图 3.2 所示。

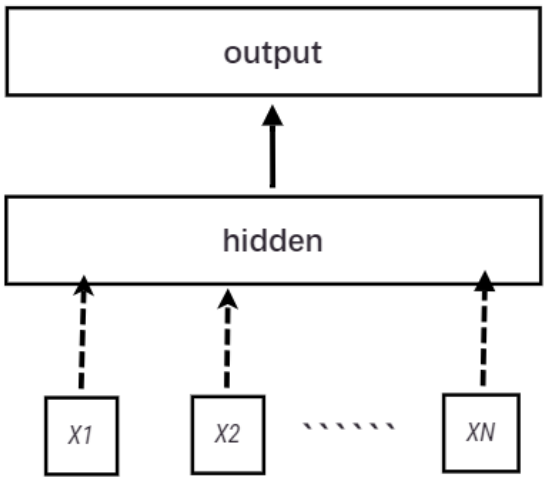



图 3.2 FastText 结构示意图

图 3.3 展示了 FastText 词向量训练过程，在输入层将输入的多个文本词语转换为向量并加入了 n-gram 特征，使语义信息更加丰富完整。隐含层是对多个词向量叠加平均，并在输出层采用树形的层次 Softmax 来替代标准的 Softmax，大大提高计算效率。

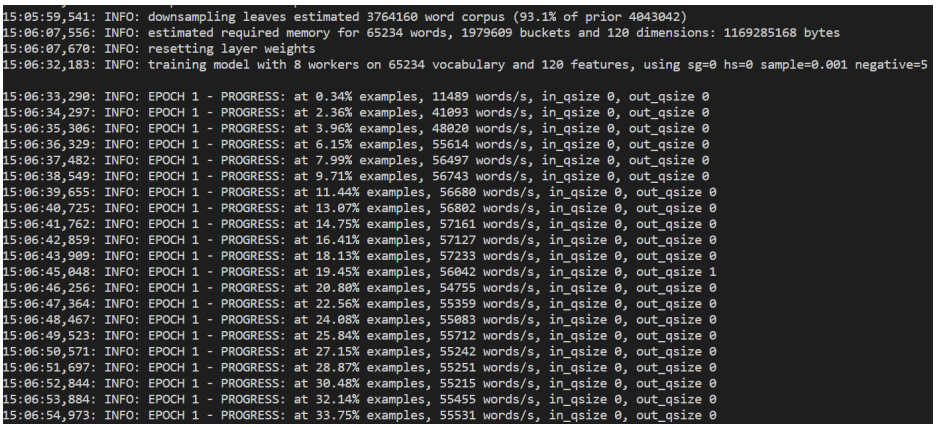



图 3.3 FastText 词向量训练过程

### 3.1.2 模型改进

FastText 模型 [4] 在输入层采用的 n-gram [8] 处理，以便学得更多的词序特征。但在中文文本的处理上，n-gram 处理后生成的词典会产生大量无意义的冗余词条。如图 3.4，在改进的 FastTest 模型中增加了筛选层删除这些无意义词，使得词语之间关系更加紧密，特征更加清晰。

短文本使用词袋模型将短文本中的词看作是一袋子单词，不考虑其语法结构和词序关系，每个词都是相互独立的。再将 n-gram 处理后的词集与短文本词袋模型中的词条进行对比，将 n-gram 处理后产生的词条但在短文本词袋模型中却不存在的词条删除。例如 “今天你敲代码了吗”在做 3-garm 处理时会产生“今天你”，“天你敲”，“你敲代”，“敲代码”，“码了吗”。此种例如“天你敲”这样的词条是无意义的，需要被筛选掉。从而保留“今天你”，“敲代码”。输入词序列 3-gram 处理过程则学习到了“今天你-敲代码”这个词与词之间的前后关系以及词语之间的语义联系特征。同时在做 n-gram 处理时。1-gram 由 n 个词组成，2-gram 由 n-1 个词组成，以此类推。那么 n-gram 模型的复杂度就为$\frac{n(n+1)}{2}$。经过无意义词去除处理可以在某种程度上减少计算量，增加运行速度。

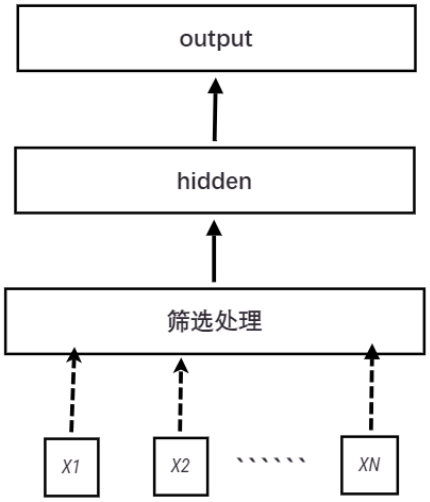



图 3.4 改进后 FastText 结构示意图

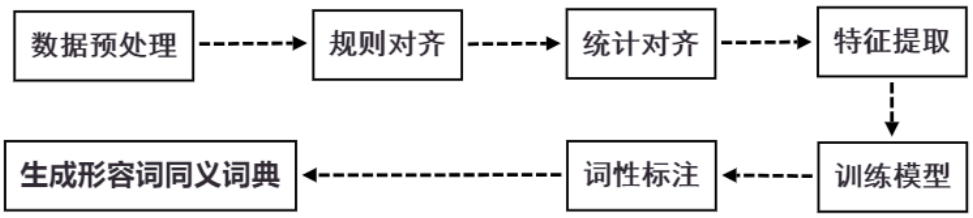
## 3.2 基于搜索 query 分析的同义词挖掘模块

由于网络上现有的同义词词典具有片面，偏差，局限等缺点，系统采用同义词挖掘的方式构建适用于某一行业领域（以物流领域为例）的同义词词典。

### 3.2.1 算法方案

2002 年由 G.Jeh 和 J.Widom 提出 SimRank 思想 [9]，基本思想在于关联到相似事物的两个事物相似。

基于这一思想，我们可以认为，点击到同一网页的所有用户 query 相似。即不同用户使用不同的搜索 query 在搜索引擎上搜索时点进了同一个网页，那么我们认为这些不同的搜索 query 的意义是相同的。从这些 query 中筛选出同义词具有包容性更强，范围更广，时代性更强等特征。因此，系统基于搜索 query 分析按照图 3.5 的流程进行同义词挖掘。




3.5 同义词挖掘流程设计图

#### 3.2.1.1 数据预处理

**1.Query 资源的数据提取**

基于 SimRank 的聚合思想 [10]，从数据库中提取出文本原始数据。考虑到用户错点行为，提取搜索词时，将前几个搜索量加和占这篇网页搜索量 80%的搜索词提取出来，并分词去掉标点。

**2.Query 搜索词两两配对**

基于 SimRank 思想，点进同一网页的搜索 query 是两两相似的。所挖掘的同义词就在两个不同的相似 query 中。

#### 3.2.1.2 规则对齐

采用机器翻译中的对齐概念——两条相似的 query 表示的是同一个意思，可以认为它们之间是可以相互翻译的。所以，理论上只要找到这两条 query 中词与词之间的对齐，进行标注，最终就可以提取出相互翻译匹配的词对并在全量数据中统计它们的系列统计特征。

通过对 query 资源数据的观察，如图 3.6 所示，将词的对齐归纳为以下四类：相同词对齐，同义词对齐，相似词对齐，结构对齐。

四类之间的优先级：相同词对齐>同义词对齐>相似词对齐>结构对齐。即被相同词对齐标注过的词对将无法被同义词对齐标注，被同义词对齐标注过的词对将无法被相似词对齐标注，以此类推。

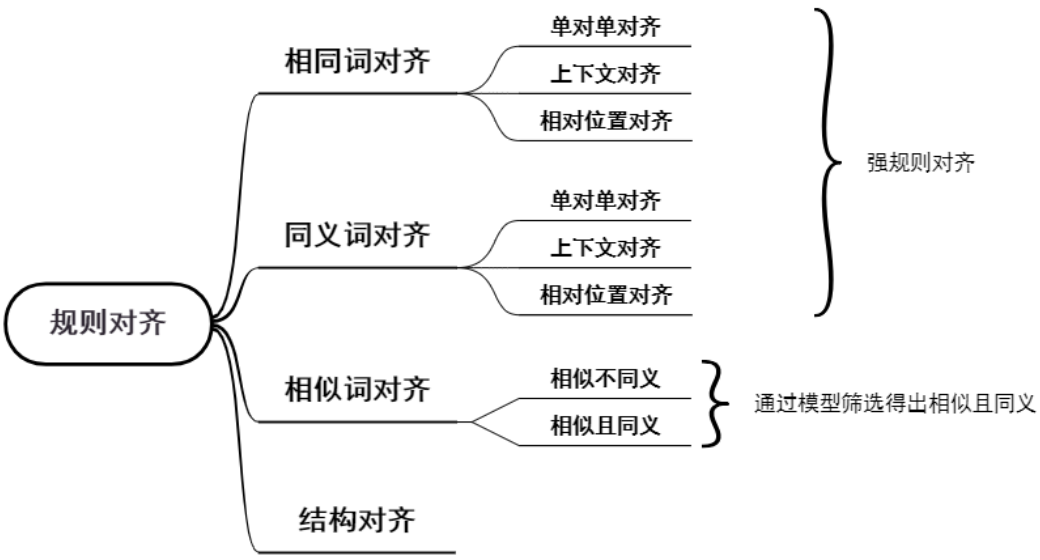



图 3.6 规则对齐结构

**1. 相同词对齐**

(1) 单对单对齐

词 W 在 query1 和 query2 中都各只有一个，那么直接可以对两个词进行相同词标注

例：“学好 数学 小技巧”，“如何 掌握 数学 学习 方法”中，两个“数学”直接进行相同词对齐标注

(2) 上下文对齐

在相同词对单过后就要进行上下文信息对齐。上下文对齐针对多词情况主要考虑上下文语境和语序。

当词 W1 在 query1 中的前一个位置或者后一个位置存在被对齐的情况，则找到它前一个词或后一个词在 query2 中的对齐位置 pos1，W1 优先于 query2 中按照先后顺序于 pos1 最接近的 query2 中相同的词 W2 对齐。

例：“舒服的 牛皮 座椅 真皮 套垫”，“牛皮 套子 正宗的 牛皮 座椅”中，座椅和座椅已经对齐，根据上下文对齐规则，query1 中的牛皮将和 query2 中第二个牛皮对齐，并且符合原意。

(3) 相对位置对齐

相对位置对齐主要解决相同词一对多或者多对多的问题。如果词 W1 在 query1 和 query2 中出现多次，将采用相对位置对齐，计算公式如下：

$$\text{pos} = \frac{(\text{j} + 1)}{\text{phrasesize}}$$

表示当前位置。两个相对位置差最小的词与相互对齐，每对齐一个词都要进行一次上下文对齐看是否有新对齐才能继续对齐。

例：“困了 喝 什么 解乏”，“喝 什么 提神 喝 甜的”，query1 中的“喝”应该与 query2 中的第一个“喝”对齐，符合语序。

**2. 同义词对齐**

同义词对齐主要是通过使用同义词典进行查询。与相同词对齐相同采用以下三种步骤：

(1) 同义词单对单对齐；

(2) 上下文信息对齐；

(3) 相对位置对齐；

和相同词对齐一样，这两种对其都被看作强规则对齐，其所标注的词对在基于统计翻译模型的迭代计算中不会相互计算一次概率，且统计对齐标注无法覆盖相同词对其标注和同一次对齐标注。

**3. 相似词对齐**

有一些字面相似的词语对，如<“飞机”，“飞鸟”>，<“鸽子”，“白鸽”>之类有构成同义词的可能，相似词对齐则将这种情况筛选出来。后续通过模型训练判别，<“飞机”，“飞鸟”>在模型中会被判错，而<“鸽子”，“白鸽”>这对正确的同义词会被判对，从而保证同义词覆盖率。

**4. 结构对齐**

当 query1 中某个词 W1 的前一个词和后一个词都分别和 query2 中某个词 W2 的前一个词和后一个词对齐了，那么 W1 和 W2 具有相同的语境，是有成为同义词的可能的，将 W1 和 W2 进行对齐标注，称作结构对齐。

例：“动脉 粥样 硬化”和“动脉 很粘稠 硬化”中。动脉和硬化都已经对齐，那么“粥样”和“很粘稠”进行结构对齐。

#### 3.2.1.3 统计对齐

统计对齐主要基于机器翻译模型 IBM [11]，但与统计翻译模型不同。该模块只采用了模型中间变量，即词语和词语之间的对齐概率，而不需要两个 query 句子之间的翻译概率。

**1. 统计翻译模型**

IBM1 模型的基本假设如下：

(1) 假设所有翻译长度都是等概率的

(2)假设词语对齐只与源语言词语有关，与其他因素无关

(3)假设目标词语的选择只与其对应的源语言词语有关，与其他因素无关

由以上假设，对齐概率公式可表示为：

$$P(F,A|E)=\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}t(f_j|e_{a_j})$$

对所有可能的对齐求和，其翻译概率公式可以表示为：

$$P(F|E)=\sum_{A}P(F,A|E)=\frac{\varepsilon}{(l+1)^m}\sum_{a_1=1}^{l}\cdots\sum_{a_m=1}^{l}\prod_{j=1}^{m}t(f_j|e_{a_j})$$

$$=\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}l\sum_{i=1}^{l}t(f_j|e_i)$$

因此，只要有单词概率分布$t(f_j|e_{a_j})$，就能计算句子翻译句子 F 的概率，并且有约束条件$\sum_f t(f|e)=1$，最终得到公式：

$$P(f|e)=\lambda_e^{-1}\sum_{A}P(F,A|E)\sum_{j=1}^{m}\delta(f,f_j)\,\delta(e,e_{a_j})$$

定义在 E 和 F 所有可能的对齐 A 下的 e 和 f 连续数的期望公式如下：

$$c(f|e,F,E)=\sum_{A}P(A|F,E)\sum_{j=1}^{m}\delta(f,f_j)\,\delta(e,e_{a_j})$$

代入迭代公式，并将$P(F\mid E)$并入参数$\lambda_e$，最终可得公式如下：

$$t(f|e)=\lambda_e^{-1}c(f|e,F,E)$$

这就是基于 IBM1 模型的统计条件概率计算公式，即对齐概率。

**2. 统计对齐步骤**

条件概率计算主要分为以下三个过程：

(1) 计算条件概率，初始条件概率使用均值;

(2) 计算条件概率的期望值;

(3) 期望值归一化，再转过程 1 计算条件概率直至条件概率收敛。

迭代完成后，如果 query1 中的一个词 W 未被对齐标注，则找到 query2 中与词 W 最大的翻译概率词 M。如果 M 是强规则对齐词语，则 W 不进行对齐操作，否则将覆盖其他对齐标注。

#### 3.2.1.4 特征提取

统计特征主要根据候选同义词对的共现次数和配对簇来构造。一个候选同义

词对，在全量数据的抽取词对过程中出现的次数越多，可认为它们之间的关联度越大，是同义词的概率也越大。根据先验知识，采用了 7 维的统计特征，如图 3.7 所示。

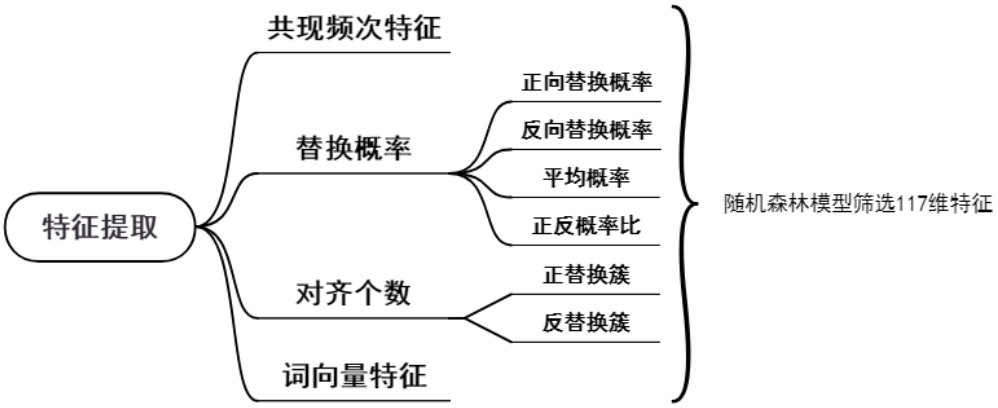



图 3.7 特征提取

**1. 共现频次特征**

共现频次就是指在进行词对抽取时，该词对出现过的次数。由先验知识可知，共现频次和同义词倾向是正相关的。采用以下公示来计算一组候选词对的共现频次特征值：

$$\frac{\text{count}}{\text{max\_count}}$$

count是这一组候选对的共现频次，max_count是所有候选词对共现频次的最大值，以此公式实现共现频次的归一化处理，将特征值限制在(0，1]之间，用来避免量纲的影响。

**2. 替换概率**

假如词 W 和词 M 是提取出的一个候选词对，词 W 可能还有上百个不同的词组成了候选词对。考虑词 W 和 M 可替换的概率，用以下公式计算替换概率的值，其中count至这一对候选词的共现信息，count_all是候选对中某一个词的所有配对的总次数。

$$\frac{\text{P} = \text{count}}{\text{count\_all}}$$

由于count_all是某一个词的共现概率，替换概率有两个，分别为以下两维特征：

(1) 正向替换概率

(2) 反向替换概率

综合考虑正反向替换概率，还提出以如下复合特征，p1 表示正向替换概率，p2 表示反向替换概率：

(1) 平均概率：

$$\frac{(p_1 + p_2)}{2}$$

(2) 正反向概率比：

$$\frac{p_1}{p_2}$$

由此，替换概率提出的四维特征：正向替换概率，反向替换概率，平均概率，正反向概率比。

**3. 对齐个数**

与词 W 有候选配对的词越多，即其对齐个数越多，是小簇对齐的概率越小，它的翻译可信度就越高。通过以下公式计算对齐特征：

$$P = align\_num/max\_align\_num$$

其中，align_num表示这一个短语对的共现可替换簇的大小，max_align_num表示全局候选数据中可替换簇的大小的最大值，也是进行了归一化的处理。一共可提出正反替换簇两维特征。以上基于先验知识，共提出了 7 维特征，从主观上理解同义词对的特征表现。

**4. 词向量特征**

除了主观基于短语分析的特征统计之外，影响同义词的判别还有一些难以描述的隐形特征。所以采用构建词向量的方式，构建 200 维度的特征表征。将它各种可描述的、不可描述的特征分散到每一维特征当中。

词向量的构建主要采用 FastText 模型 [4]。设置参数 200 维度，并添加上由先验知识得出的 7 维特征。经过随机森林模型对特征进行筛选，利用随机森林对特征进行扰动和采样机制对特征重要性进行计算。实验证明筛选出的 117 维特征是效果最好的，根据观察，之前由先验知识提出的 7 维特征在筛选过后

依旧被保留。

#### 3.2.1.5 训练模型

选用逻辑回归对模型进行训练。由于逻辑回归是有监督模型，所以需要人工进行同义词对打标签。

由于提取出的候选对，绝大多数都不是同义词。所以采用 1 个 1000 条正样例和 5 个 1000 条反样例组合成 5 个数据集进行训练。

设置迭代次数为 1000 次，最终训练好了逻辑回归模型进行判断同义词对，初步建立同义词对词典。

#### 3.2.1.6 词性标注

考虑到会要处理专业相关的名词解释和概念辨析题。其中专有名词一般没有同义词，不可被替代。故要在同义词库中筛选出形容词和动词作为题目所用的同义词库。利用 HanLP 库对模型判别后的同义词对进行词性标注，并将形容词和动词筛选出来。

## 3.3 基于改进 TextRank 的关键词提取模块

在进行关键词的提取时主要采用了以下两种算法：TF－IDF 算法 [12] 和平均信息熵算法 [13]。前者主要用于词语对单个文档，而后者主要用于词语对文档集的重要性计算。

### 3.3.1 算法介绍

#### 3.3.1.1 TF－IDF 算法

TF：词频（term frequency）

词频指的是某一个指定词在一篇文档中出现的次数，为了避免词频偏向长文档（同一个词可能在长文档里比短文档里出现的次数多，而不管重要与否），所以用词出现的次数比文档的总词数作为归一化公式以防止它偏向长的文章。

$$\mathrm{TF} = \frac{\text{在某个文档中词条出现的次数}}{\text{该文档的所有词条数目}}$$

IDF：逆文档频率(inverse document frequency)

有些通用词在每个文档中都会大量出现，比如‘的’这样的词，用 TF 公式计算出来的权重肯定很大，但是这样的词无法反应一篇文档的主题，需要那些在一篇文档中出现的多而在其他文档中出现的少的词，这一类的词才能反映文档主题，显然 TF 是做不到这一点的，而逆向文件频率恰好可以做到这一点。

IDF 的思想：如果包含某个词条的文档越少，IDF 越大，则说明词条具有很好的类别区分能力。按如下公式计算某个词条的 IDF 值：

$$\mathrm{IDF} = \log \frac{\text{语料库中文档总数}}{\text{包含指定词条的文档数}+1}$$

（注：公式中分母加 1 是为了防止分母是 0。）

TF－IDF 算法的基本思想是：利用词频和逆文档频率相乘得到词语的权重值。根据 TF－IDF 算法，词语权重$W_{TF\text{-}IDF}(\mathrm{i})$ 的计算公式如下：

$$\mathrm{W_{TF-IDF}}(\mathrm{i}) = \mathrm{TF_i} * \mathrm{IDF_i}$$

由上述公式可得，如果词语在某篇文档中出现频率较高，但是在语料库中包含该词的文档数较低，则该词根据 TF－IDF 算法得到的权重值$\mathrm{W_{TF-IDF}}$（i）就越高，就认为该词可以一定程度上反映出文章的主题内容。反之，说明该词语不是重要的，不能够反映出主题。

#### 3.3.1.2 平均信息熵算法

平均信息熵的基本思想是: 根据词频在不同文章中出现的频数，利用整体语料库计算所有词语分别对于单个文档和文档集这两种情况的重要性，通过平均信息熵可以衡量 词语在整个文档集中分布的均衡度。根据平均信息熵算法，词语权重$\mathrm{W_{Entropy}(i)}$ 的计算公式如下:

$$\mathrm{W_{Entropy}(i)} = 1 - \frac{1}{\log \mathrm{N}} \sum_{\mathrm{k=1}}^{\mathrm{N}} \left(\frac{\mathrm{f_{wk}}}{\mathrm{n_w}} \log \frac{\mathrm{n_w}}{\mathrm{f_{wk}}}\right)$$

其中，$\mathrm{f}_{wk}$ 表示词 w 在文档 k 中出现的频次，$\mathrm{n_w}$ 表示词 w 在整个文档

集中出现的频次，N 表示语料库中文档的总数。

如果词 i 在各类别文档中出现频率相当，则其$W_{Entropy}(i)$的值接近于最小值 0，表明不是很重要，不能表现文章主题。反之，如果词语 i 在不用文档中出现的次数差别很大时，其$W_{Entropy}(i)$的值接近于最大值 1，表示这个词能够很好的反映主题。

### 3.3.2 关键词提取实现

#### 3.3.2.1 算法描述

传统 TextRank 算法 [3] 将一篇文档转换成一张有向带权的词图模型，是将文本进行分割，分割成基本单元，即词语，每个基本单元看作是一个节点，每个节点之间的边由词节点之间的共现关系决定，而节点的重要性又由相邻节点指向数量决定。TextRank 算法的计算方式如下所示。

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{W_{ji}}{\sum_{V_k \in Out(V_j)} W_{jk}} WS(V_j)$$

构建 TextRank 关键词图 G=（V，E），其中 V 为节点集合，E 为节点之间的边集合；$In(V_i)$ 是节点$V_i$ 的入度点的集合，即指向节点$V_i$的节点集合;$Out(V_j)$ 是节点$V_j$的出度点集合，即节点$V_j$指向的所有节点的集合;$W_{ji}$ 是$V_j$节点与节点$V_i$之间边的权重;d 是阻尼系数，一般取值为 0.85，其作用是表示当前节点向其它任意节点跳转的概率，同时能够保证让权重能够稳定的传递至收敛，最终计算每个词语的权重并进行排序，选取 topN 作为 N 个关键词并输出。在这里对$W_{ji}$权重进行改进。选取任意一个词 i，定义词 i 的综合权重计算方式 如下:

$$W_{Weight}(i) = \frac{1}{2} W_{TF-IDF}(i) + \frac{1}{2} W_{Entropy}(i)$$

其中，$W_{TF-IDF}$（i）是词语通过 TF－IDF 计算得到的权重值，$W_{Entropy}(i)$是词语的平均信息熵权值。根据 TextRank 算法，节点间的转移概率计算公式为:

$$W（V_j, V_i）= \frac{W_{Weight}(V_i)}{\sum_{V_k \in Out(V_j)} W_{Weight}(V_k)}$$

根据上述公式，节点的权重迭代公式如下:

$$WS（V_i）= (1-d) + d * \sum_{V_j \in In(V_i)} W(V_j, V_i) WS(V_j)$$

其中，$W（V_j, V_i）$表示节点$V_j$到节点$V_i$边的转移概率，用上述给出的转移概率计算公式得到；对于$W_{Weight}(V_i)$即前述的综合权重值。

#### 3.3.2.2 关键词提取

基于改进的 TextRank 关键词提取算法的提取流程如图 3.8 所示。

首先将需要提取关键词的文本进行输入，接着进行三个模块处理：

(1) 模块一：文本预处理。对文本中的内容进行分词，词性标注，只保留名词、专有名词、动词、形容词和副词，对文本中的提用词进行删除。

(2) 模块二：权重计算。用上述的两个算法计算得到文本中每个词语的$W_{TF-IDF}（i）$和$W_{Entropy}(i)$，并计算综合权重$W_{Weight}(V_i)$。

(3) 模块三：提取关键词。构建基于词语综合权重的加权节点初始值及节点概率转移矩阵改进的 TextRank 模型，最终计算选择前 N 个权重比较大的词语作为关键词并输出。

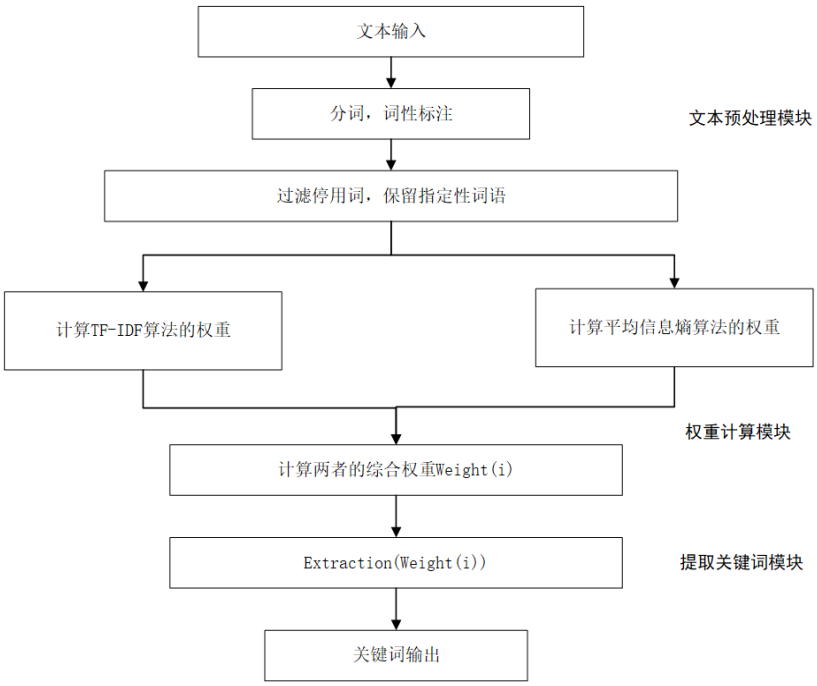



图 3.8 关键词提取算法流程图

# 4 实验结果

使用了包含 100 个样例的测试集进行测试，最终准确率为 94%。下面展示测试集中能从不同角度反应系统功能与性能的六个样例：

(1) 系统样例一：近义词的处理

该系统针对之前调研结果提出的近义词判别问题，借助 FastText 技术和调用 synonyms 库很好的解决了这个问题。效果如图 4.2 所示；

期望得分：10　　实际得分：10

近义词对：　选择出的——找到的　　反映——体现


图 4.1 “近义词处理类”样例测试

(2) 样例二：缺少得分点的处理

学生答案相对于标准答案残缺不全是这类题型中最为普遍的现象之一，该系统能对此很好的识别与判分。效果如图 4.2；

期望得分：6　　实际得分：6

缺少得分点：“传递性”、“不对称性”


图 4.2“缺少得分点处理类”样例测试

(3) 样例三：反义词、否定词与局部否定的处理

该系统也可以对反义词或否定词进行判别，并可以同时对所有涉及的类型进行判别。效果如图 4.3；

期望得分：5　　实际得分：5

否定（反义）词对：客观性——主观性　不对称性——对称性


图 4.3 “反义词、否定词与局部否定处理类”样例测试

(4) 样例四：非关键性成分的处理

对于具体阐释性问题，尤其是较长文本的回答，会有一些非关键性的、非必要的成分，而在有限考试时间内许多考生会精简甚至舍弃这部分。本系统对该类答案亦有较好的识别与处理。效果如图 4.4；

期望得分：6　　实际得分：6

该学生答案不仅在得分点上有出入，还在不影响语义的前提下删减了非必要成分。


图 4.4 “非关键性成分的处理类”样例测试

(5) 样例五：长文本中大面积否定的处理

该系统也能对长文本中的否定语义进行准确识别，不是简单的判别单个否定词语和局部否定。效果如图 4.5；

期望得分：0　　实际得分：0

对于标准答案中的肯定或者否定，学生答案全部为对应的否定。


图 4.5 “长文本中大面积否定的处理类” 样例测试

(6) 样例六：行业专业术语及专有名词的处理

基于训练所用的物流专业领域的语料库，以物流领域为例对行业专业术语概念释义和名词解释类主观题进行测试，普遍存在大量的不可拆分的专业术语或专有名词，对于这种情况，普通的文本相似度计算方案不能适用，但本系统的机制能对此有较好的处理。效果如图 4.6；

期望得分：5　　实际得分：5


图 4.6 “行业专业术语及专有名词的处理类” 样例测试, “分布性”、“标准性”等是物流专业术语，不可缺字错字

# 5　系统使用说明

## 5.1 应用与开发环境

本系统应用环境描述如下：Python3.6 及以上、Mysql5.6 及以上、JDK1.8 及以上、Tomcat9 及以上、Windows7 及以上。

本系统开发技术与环境如表 5.1 所示。

表 5.1 开发技术与环境

| | 开发工具 | 开发技术、库与环境 |
|---|---|---|
| 算法模块 | Jupyter Notebook;<br>Vscode1.38.1;Pyhcarm2019; | jieba；jiagu；<br>itertools；synonyms；<br>gensim |
| 网页端 | IntelliJ IDEA 2018；<br>vue-cli3.0; | JavaScript 语言；<br>Vue 框架 |
| 服务器 | MobaXterm ;<br>Tomcat9（Spring Boot 内置） | Centos7.7;<br>Windows10 |
| Java 后端 | IntelliJ IDEA 2019; | Spring Boot;<br>Maven;<br>MyBatis |
| 数据库 | Navicat;<br>PhpMyAdmin; | MySQL ; |

## 5.2 运行与使用

首先根据 5.1 节保证相关环境和软件的正确配置和安装。

算法模块代码位于“\代码\python”路径下，语料库以及词典包含在“\代码\data”路径下，go.py 为算法主程序。所需的输入格式如图 5.1。


图 5.1 算法输入格式

Web 服务端模块位于“\代码\web”路径下，按照前端的 vue 框架和后端的 spring boot 框架以及 flask 框架搭建而成，运行需要按照按照相关框架进行相关配置，比如运行 vue.js 所需的依赖，springboot 框架所需的 maven 环境，以及管理数据库所需的 mybatis 等，最后通过 IDEA，导入 maven 项目，根据 pom.xml 文件打成 jar 包，直接通过 java -jar 的方式运行即可。

# 6 总结

本文探讨了当前教育领域在主观题智能评阅方面的需求，指出传统阅卷方式存在的效率低、成本高、主观性和不确定性等问题。强调了智能主观题评阅系统的广泛应用前景，但也提出了在中文语境中的一些挑战，如词义多义、停用词无法有效识别、新词新义的更新速度等。在此背景下，本文设计了一款基于改进的 FastText 模型、同义词挖掘模块和 TextRank 算法的主观题智能评阅系统，通过多角度、全方位的分析，提供了可推广的长文本语义相似度计算方案。

该系统不仅能够分析句子结构、识别否定词，还成功处理了动词、形容词、副词等的双重否定或反义词与原词的相似性匹配。此外，它还具备处理解释性语句，实现词语与其解释之间匹配的能力。在评分方面，系统实现了根据回答中命中关键词并符合答案意义的方式进行分数分配，超越了传统方法。

总体而言，本文提出的智能主观题评阅系统结合了改进的 FastText 模型计算相似度模块、搜索 query 分析的同义词挖掘模块和 TextRank 算法的关键词提取模块等关键技术。通过使用短文本匹配和网络通信协议等多项技术，该系统具备较好的创新性、实用性与良好效果。本研究不仅在通用性上强调了其适用性，而且在处理语言细节上取得了重要进展，为提高主观题评价的准确性和效率，推动自动评分系统领域的发展提供了有力支持。系统的潜在应用范围覆盖了各种教育场景，为教育工作者提供了一个有价值的工具，以更为复杂和深刻的方式评估学生答案并提供反馈。

# 7 参考文献

[1] Zhou L, Zhang D. NLPIR: A theoretical framework for applying natural language processing to information retrieval[J]. Journal of the American Society for Information Science and Technology, 2003, 54(2): 115-123.

[2] W. Che, Y. Feng, L. Qin, T. Liu, N-LTP: An open-source neural chinese language technology platform with pretrained models, in: Proc. 2021 Conf.

Empir. Methods Nat. Lang. Process., Association for Computational Linguistics, 2021: pp. 42–49.

[3] Mihalcea R, Tarau P. Textrank: Bringing order into text[C].Proceedings of the 2004 conference on empirical methods in natural language processing. 2004: 404-411.

[4] Joulin A, Grave E, Bojanowski P, et al. Fasttext. zip: Compressing text classification models[J]. arXiv preprint arXiv:1612.03651, 2016.

[5] Filipova O. Learning Vue. js 2[M]. Packt Publishing Ltd, 2016.

[6] Cheng T, Lauw H W, Paparizos S. Entity synonyms for structured web search[J]. IEEE transactions on knowledge and data engineering, 2011, 24(10): 1862-1875.

[7] Jiang M, Liang Y, Feng X, et al. Text classification based on deep belief network and softmax regression[J]. Neural Computing and Applications, 2018, 29: 61-70.

[8] Kim J Y, Shawe-Taylor J. Fast string matching using an n-gram algorithm[J]. Software: Practice and Experience, 1994, 24(1): 79-88.

[9] Jatnika D, Bijaksana M A, Suryani A A. Word2vec model analysis for semantic similarities in english words[J]. Procedia Computer Science, 2019, 157: 160-167.

[10] Jeh G, Widom J. Simrank: a measure of structural-context similarity[C].Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002: 538-543.

[11] Moore R C. Improving IBM word alignment model 1[C]. Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04). 2004: 518-525

[12] Liu C, Sheng Y, Wei Z, et al. Research of text classification based on improved TF-IDF algorithm[C]. 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE). IEEE, 2018: 218-222.

[13] Cover T M, Thomas J A. Entropy, relative entropy and mutual information[J]. Elements of information theory, 1991, 2(1): 12-13.