目录

第二届搜狗“短文本语义相关度计算”竞赛——算法的说明文档

零：团队信息：

团队用户名：jacoxu@126.com；团队名：CBrain_xtz

团队成员：许家铭，田俊，周世玉；指导老师：徐博，田冠华

编程语言：Java，Matlab

一：数据分析：

标注数据：22591；未标注数据：13287，原始文件编码格式 GB2312

标注得分 0、1、2、3，共 4 个得分，且得分比为：

3:2:1:0 = 2290:8288:8284:3729 = 10.14 : 36.69 : 36.67 : 16.51

二：算法说明：

本文中所描述的短文本语义相关度计算框架图如图 1 所示：

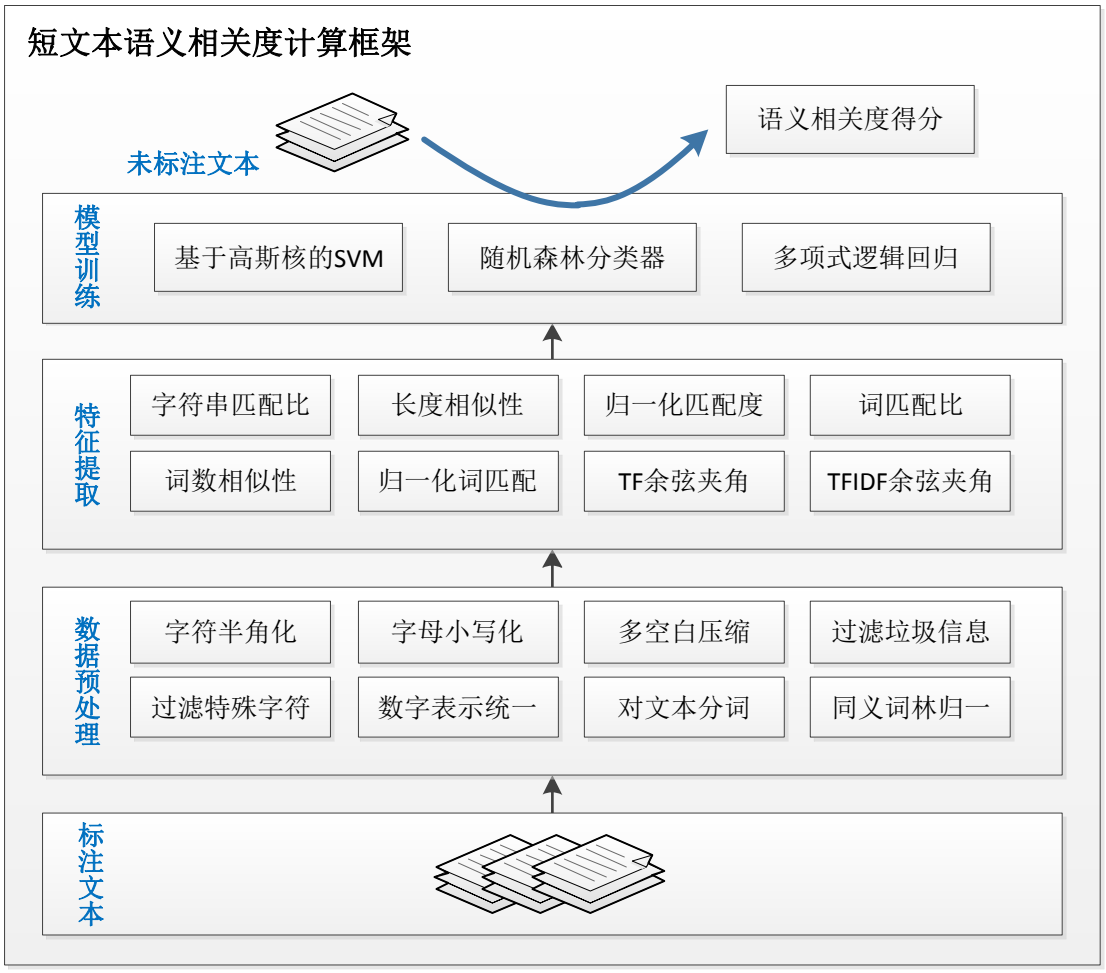

图 1. 短文本语义相关度计算框架

2.1 数据预处理

a). 特征归一化

（注，需要对发布的文本文件先转成 UTF-8 格式，方便跨平台乱码问题）

1. 全角转半角；

预处理标注数据：2015 短文本语义相关度比赛标注数据集.txt

预处理未标注数据：2015 短文本语义相关度比赛评测数据集.txt

```
%java code
tmpQueryStr=Full2Half.ToDBC(tmpQueryStr);//2.1.a.1-全角转半角
tmpTitleStr=Full2Half.ToDBC(tmpTitleStr);//2.1.a.1-全角转半角
```

2. 字母全部小写；

```
%java code
tmpQueryStr=tmpQueryStr.toLowerCase();//2.1.a.2-字母全部小写
tmpTitleStr=tmpTitleStr.toLowerCase();//2.1.a.2-字母全部小写
```

3. 多个空格缩成单个空格；

```
%java code
tmpQueryStr=tmpQueryStr.replaceAll("\\s+", " ");//2.1.a.3-多个空格缩成单个空格
tmpTitleStr=tmpTitleStr.replaceAll("\\s+", " ");//2.1.a.3-多个空格缩成单个空格
```

4. 过滤结果出处、信息来源等垃圾无效信息；

初步过滤规则：首先利用“-”或“_”对 Title 切分，强制保留第一切分块，并对后面切分块进行信息冗余判断，判断规则为，直接对 Query 和 Title 拆分二元词，然后匹配是否存在共现词，若无，则判断为冗余信息过滤，若有则保留。

```
%java code
String[] tmpTitleArray= tmpTitleStr.split(" - |—|-|_");//2.1.a.4-拆分后判断可疑垃圾冗余信息
for (int i = 0; i < tmpTitleArray.length; i++) {
    String tmpQuerySnippet = tmpTitleArray[i].trim();
    if (tmpQuerySnippet.length()<1) continue;
    if (hasContent) {
        //判断是不是 IT 后缀垃圾无效信息
        if (isITSuffixSpamInfo(query2TermSet, tmpQuerySnippet)){
            continue;
        }else {
            tmpContentBuffer.append(" "+tmpQuerySnippet);
        }
    }else {
        //如果还没有内容，则不需要判断是否垃圾，直接写入到 Content 中
        tmpContentBuffer.append(tmpQuerySnippet);
        hasContent = true;
    }
}

private boolean isITSuffixSpamInfo(HashSet<String> query2TermSet, String tmpQuerySnippet) {
    String[] tmpQueryTerms = SplitNGram.split2NGram(tmpQuerySnippet, 2).split(" ");
    String queryTermStr;
    for (int i = 0; i < tmpQueryTerms.length; i++) {
        queryTermStr = tmpQueryTerms[i].trim();
        if ((queryTermStr.length()>0)&&query2TermSet.contains(queryTermStr))
            return false;
    }
    return true;
```

```
}
```

5. 过滤掉特殊字符，并以空格隔开；

过滤规则 1：生成特殊字符库

过滤规则 2：仅保留中文汉字、英文字母及数字

```
%java code
tmpQueryStr = StringAnalyzer.extractGoodCharacter(tmpQueryStr);
tmpTitleStr = StringAnalyzer.extractGoodCharacter(tmpTitleStr);

public static String extractGoodCharacter(String ss){
	if(ss == null)
		return null;
	Boolean lastCharTag = true;
	StringBuffer str = new StringBuffer();
	char[] ch = ss.toCharArray();
	for (int i = 0; i < ch.length; i++) {
		if(CharacterAnalyzer.isGoodCharacter(ch[i])){
			str.append(ch[i]);
		}else {
			str.append(' ');
		}
	}
	return str.toString().replaceAll("\\s+", " ").trim();
}

public static final boolean isGoodCharacter(char c)
{
	if(filterUnicode(c))
		return false;
	if(isChinese(c))
		return true;
	if(isWhiteSpace(c))
		return true;
	if(isNumber(c))
		return true;
	if(isEnglish(c))
		return true;
	return false;
}
```

6. 数字表示统一；

将文本中所有数字转统一转换成阿拉伯数字

如“一”和“壹”，均表示为“1”。

```
%java code
lineTXT = lineTXT.replaceAll("零", "0");
lineTXT = lineTXT.replaceAll("壹|一", "1");
```

```
lineTXT = lineTXT.replaceAll("贰|二", "2");
lineTXT = lineTXT.replaceAll("叁|三", "3");
lineTXT = lineTXT.replaceAll("肆|四", "4");
lineTXT = lineTXT.replaceAll("伍|五", "5");
lineTXT = lineTXT.replaceAll("陆|六", "6");
lineTXT = lineTXT.replaceAll("柒|七", "7");
lineTXT = lineTXT.replaceAll("捌|八", "8");
lineTXT = lineTXT.replaceAll("玖|九", "9");
```

7. 对 Query 和 Title 进行分词

分词工具 Java 版的 ansj

```
%java code
tmpQueryTermList = DivideWord.splitWord(tmpQueryStr);
tmpTitleTermList = DivideWord.splitWord(tmpTitleStr);

public class DivideWord {
    public static ArrayList<String> splitWord(String shortDoc) throws IOException {
        ArrayList<String> termsOnDoc = new ArrayList<String>();
        List<Term> all = new ArrayList<Term>();
        Analysis udf = new ToAnalysis(new StringReader(shortDoc));
        Term term = null;
        while ((term = udf.next()) != null) {
            String tempTerm = term.toString().trim();
            if (tempTerm.length()>0){
                termsOnDoc.add(tempTerm);
            }
        }
        return termsOnDoc;
    }
}
```

8. 进行同义词林归一化

同义词转化词典：哈工大社会计算与信息检索研究中心同义词词林扩展版

```
%java code
String[] tmpQueryTermList = tmpQueryStr.split("\\s+");
String[] tmpTitleTermList = tmpTitleStr.split("\\s+");
//对 Query 的分词结果进行同义词归一
for (int i = 0; i < tmpQueryTermList.length; i++) {
    if (i!=0) {
        tmpContentBuffer.append(" ");
    }
    //如果同义词典里面有此词的话，则替换掉
    if (SmsBase.getsynWordsMap().containsKey(tmpQueryTermList[i])) {
        tmpContentBuffer.append(SmsBase.getsynWordsMap().get(tmpQueryTermList[i]));
    }else {
        tmpContentBuffer.append(tmpQueryTermList[i]);
```

```
        }
    }
    tmpContentBuffer.append("\t");
    //对 Title 的分词结果进行同义词归一
    for (int i = 0; i < tmpTitleTermList.length; i++) {
        if (i!=0) {
            tmpContentBuffer.append(" ");
        }
        //如果同义词典里面有此词的话，则替换掉
        if (SmsBase.getsynWordsMap().containsKey(tmpTitleTermList[i])) {
            tmpContentBuffer.append(SmsBase.getsynWordsMap().get(tmpTitleTermList[i]));
        }else {
            tmpContentBuffer.append(tmpTitleTermList[i]);
        }
    }
```

b). 特征提取

（如下部分特征分别在未进行同义词转换及进行同义词转换的数据上提取特征 2 份）

1. 去掉空格判断 Title 是否完全匹配包含 Query 字符串 （同义词归一前版本）

```
%java code
//先去掉所有的空格
tmpQueryStr = tmpQueryStr.replaceAll("\\s+", "");
tmpTitleStr = tmpTitleStr.replaceAll("\\s+", "");
boolean tContainsQ = tmpTitleStr.contains(tmpQueryStr);
if (tContainsQ) {
    tmpFeaBuffer.append("1");
}else {
    tmpFeaBuffer.append("0");
}
```

2. 去掉空格判断 Title 与 Query 长度之间的相似性：（同义词归一前版本）

长度相似性 = 1 – 长度差的绝对值/较长文本长度

```
%java code
//先去掉所有的空格
tmpQueryStr = tmpQueryStr.replaceAll("\\s+", "");
tmpTitleStr = tmpTitleStr.replaceAll("\\s+", "");
int tmpQueryLen = tmpQueryStr.length();
int tmpTitleLen = tmpTitleStr.length();

//长度绝对值
int tmpAbsLenDiff = Math.abs(tmpQueryLen - tmpTitleLen);
double strLengthSim = 0;
//进行长度归一
if (tmpQueryLen>=tmpTitleLen) {
```

```
    strLengthSim = 1-((double)tmpAbsLenDiff/(double)tmpQueryLen);
}else {
    strLengthSim = 1-((double)tmpAbsLenDiff/(double)tmpTitleLen);
}
```

3. 字符串完全匹配值 乘以 长度相似性：（同义词归一前版本）

2.b).1 * 2.b).2

```
%java code
Result2Txt(tarFilePath,String.valueOf(tStrContainQ*strLengthSim));
```

4. 对 Query 进行分词，然后判断 Title 包含 Query 的 Term 比 （同义词归一前/后两个版本）

Title 包含 Query 的 Term 数 / Query 的 Term 总数

```
%java code
//先对 Query 进行分词，预处理阶段 a.7 以通过 ansj 分词完毕，因而只需要空格分割即可
String[] tmpTermList = tmpQueryStr.split("\\s+");
for (int i = 0; i < tmpTermList.length; i++) {
    if (tmpTitleStr.contains(tmpTermList[i])) {
        tmpTermNum=tmpTermNum+1;
    }
}
//进行归一
tmpTermNum = tmpTermNum/(double)tmpTermList.length;
```

5. 对 Title 与 Query 进行分词，判断 Terms 长度之间的相似性：（同义词归一前/后两个版本）

长度相似性 = 1 – 长度差的绝对值/较长文本长度

```
%java code
int tmpQueryLen = tmpQueryStr.split("\\s+").length;
int tmpTitleLen = tmpTitleStr.split("\\s+").length;

//长度绝对值
int tmpAbsLenDiff = Math.abs(tmpQueryLen - tmpTitleLen);
double strLengthSim = 0;
//进行长度归一
if (tmpQueryLen>=tmpTitleLen) {
    strLengthSim = 1-((double)tmpAbsLenDiff/(double)tmpQueryLen);
}else {
    strLengthSim = 1-((double)tmpAbsLenDiff/(double)tmpTitleLen);
}
```

6. Title 包含 Query 的 Term 比 乘以 长度相似性：（同义词归一前/后两个版本）

2.b).4 * 2.b).5

```
%java code
Result2Txt(tarFilePath,String.valueOf(tmpTermNum*strLengthSim));
```

7. 创建词典，并生成向量空间模型 VSM；（同义词归一前/后两个版本）【无特征】

同义词归一化之前有 31001 个词，归一化之后有 26260 个词

将标注数据和未标注数据同时读取以创建公用词典，并生成空间模型 VSM

8. 导入到 Matlab 中，生成.mat 格式特征数据集合；（同义词归一前/后两个版本）【无特征】

9. 基于 TF 的夹角余弦；（同义词归一前/后两个版本）

计算所有 Query 和其对应 Title 基于 TF 值的夹角余弦

```
%matlab code
%计算基于 TF 的夹角余弦值
%先进行正则化
testQuery_TF = normalize(testQuery_TF);
testTitle_TF = normalize(testTitle_TF);
trainQuery_TF = normalize(trainQuery_TF);
trainTitle_TF = normalize(trainTitle_TF);

for i=1:length(testQuery_TF(:,1));
    TFCosineSim_test(i) = testQuery_TF(i,:)*testTitle_TF(i,:)';
    if (mod(i,1000)==0)
        disp(['hasProcessed text numbers:',num2str(i)])
    end
end

for i=1:length(trainQuery_TF(:,1));
    TFCosineSim_train(i) = trainQuery_TF(i,:)*trainTitle_TF(i,:)';
    if (mod(i,1000)==0)
        disp(['hasProcessed text numbers:',num2str(i)])
    end
end
```

10. 基于 TF-IDF 的夹角余弦；（同义词归一前/后两个版本）

计算所有 Query 和其对应 Title 基于 TF-IDF 的夹角余弦

```
%matlab code
%利用 VSM 输入得到训练数据和测试数据的 TF-IDF
disp('Compute TF-IDF')
[testQuery_TFIDF,testTitle_TFIDF,trainQuery_TFIDF,trainTitle_TFIDF] ...
        = tf_idf(testQuery_TF, testTitle_TF,...
                trainQuery_TF, trainTitle_TF);

%计算基于 TF 的夹角余弦值
%先进行正则化
testQuery_TFIDF = normalize(testQuery_TFIDF);
testTitle_TFIDF = normalize(testTitle_TFIDF);
trainQuery_TFIDF = normalize(trainQuery_TFIDF);
trainTitle_TFIDF = normalize(trainTitle_TFIDF);

for i=1:length(testQuery_TFIDF(:,1))
    TFIDFCosineSim_test(i) = testQuery_TFIDF(i,:)*testTitle_TFIDF(i,:)';
```

```
    if (mod(i,1000)==0)
        disp(['hasProcessed text numbers:',num2str(i)])
    end
end

for i=1:length(trainQuery_TFIDF(:,1))
    TFIDFCosineSim_train(i) = trainQuery_TFIDF(i,:)*trainTitle_TFIDF(i,:)';
    if (mod(i,1000)==0)
        disp(['hasProcessed text numbers:',num2str(i)])
    end
end
```

特征总结：

序号	特征名	说明	取值范围
1	tStrContainQ_nonSyn	2.b).1	0 or 1
2	strLengthSim_nonSyn	2.b).2	[0-1]
3	strConLen_nonSyn	2.b).3	[0-1]
4	tStrContainQTerm_nonSyn / tStrContainQTerm_Syn	2.b).4	[0-1]
5	tStrConTermLen_nonSyn / tStrConTermLen_Syn	2.b).5	[0-1]
6	strTermsConLen_nonSyn / strTermsConLen_Syn	2.b).6	[0-1]
7	TFCosine_nonSyn / TFCosine_Syn	2.b).9	[0-1]
8	TFIDFCosine_nonSyn / TFIDFCosine_Syn	2.b).10	[0-1]

2.2 分类器模型训练

a). 抽取标注数据的相似度标签

将标注数据的相似度标签保存到./feaSet/Step2_2_a_tags.txt 文件中

b). 基于高斯核的 SVM

抽取 1/5 做为测试数据，线性 SVM 的 ACC 可以到 52%，基于高斯核的 SVM 可以到 56%，因而选择基于高斯核的 SVM 分类器。

```
%matlab code
model = svmtrain(train_labels,labeled_fea,['-q -c 1 -g ',int2str(length(labeled_fea(1,: )))]);
```

c). 随机森林分类器（Random Forest）

```
%matlab code
nTree = 1000;
model = TreeBagger(nTree, labeled_fea, train_labels);
```

d). 逻辑回归（多项式 MultiNomial logistic Regression）

```
%matlab code
model = mnrfit(labeled_fea, train_labels);
```

2.3 语义相似度预测

基于高斯核的 SVM 预测：

```
%matlab code
predict_label = svmpredict(y,unlabeled_fea,model);
```

基于随机森林分类器预测：

```
%matlab code
[predict_label,predict_scores] = predict(model, unlabeled_fea);
```

基于逻辑回归预测：

```
%matlab code
predict_scores = mnrval(model, unlabeled_fea);
```

选择最优模型的预测结果作为输出