**硕士毕业材料说明文档\_裴家欢\_20170623**

目录

[1 毕业 3](#_Toc486751511)

[00 绘图 3](#_Toc486751512)

[01 中文词汇语义相似度计算研究-Final.doc 3](#_Toc486751513)

[02 大论文-版权.PDF 3](#_Toc486751514)

[03 大论文-独创.PDF 3](#_Toc486751515)

[04 硕士答辩\_裴家欢\_20170606\_词汇相似度计算.pptx 3](#_Toc486751516)

[2 科研 3](#_Toc486751517)

[01 中文词汇语义相似度 3](#_Toc486751518)

[001 文献阅读 3](#_Toc486751519)

[002评测任务及结果 3](#_Toc486751520)

[003实验代码与数据 4](#_Toc486751521)

[004 NLPCC会议 8](#_Toc486751522)

[005 TALLIP期刊 8](#_Toc486751523)

[02 用户查询时间意图消歧 8](#_Toc486751524)

[001 文献阅读 8](#_Toc486751525)

[002 评测任务及结果 8](#_Toc486751526)

[003 实验代码与数据 8](#_Toc486751527)

[004 NTCIR会议 9](#_Toc486751528)

[005 已写未发专利 9](#_Toc486751529)

[03 小论文汇总 9](#_Toc486751530)

[001 已发表 9](#_Toc486751531)

[002 未发表 10](#_Toc486751532)

[3 基金 10](#_Toc486751533)

[01 句子相似度计算2016 10](#_Toc486751534)

[02 跨语言信息检索结项 10](#_Toc486751535)

[4 其他 10](#_Toc486751536)

[01 GPU平台搭建 10](#_Toc486751537)

[001 GPU申请 10](#_Toc486751538)

[002 电脑配置 11](#_Toc486751539)

[003 环境搭建 11](#_Toc486751540)

[02 规则修正分词工具 11](#_Toc486751541)

# 1 毕业

## 00 绘图

毕业大论文中用到所有的图片源文件。

## 01 中文词汇语义相似度计算研究-Final.doc

大论文最终版。

## 02 大论文-版权.PDF

## 03 大论文-独创.PDF

## 04 硕士答辩\_裴家欢\_20170606\_词汇相似度计算.pptx

硕士答辩ppt.

# 2 科研

## 01 中文词汇语义相似度

### 001 文献阅读

按照英文、中文、硕博论文三类整理的相关文献压缩包。

### 002评测任务及结果

|  |  |
| --- | --- |
| 文件 | 说明 |
| evagline3.pdf | NLPCC评测任务描述 |
| Evaluation results of Task 3\_7\_14.pdf | 评测排名，评价指标为Pearson |
| Evaluation results of Task 3\_7\_20.pdf | 评测排名，评价指标为Spearman |
| LNCS-Springer\_Copyright\_Form.pdf | 版权转让说明，电子表格及扫描件 |
| NLPCC-CLSC-PeiJ\_slides.pptx | 参加NLPCC会议时的ppt |
| Semeval-2012 task4 Task Description.docx | 一个类似的评价任务 |
| 重要网址.docx |  |

### 003实验代码与数据

① 环境配置

代码使用Python实现，网页演示则使用Python对应的Django框架实现。建议安装Anaconda（Python2.X）环境，Django框架可在使用命令pip install django安装。

② 代码功能

代码整体结构如图所示，其中Com, Counter, Eval, PreProc, Single是相似度计算的主体程序，data是存放代码所用数据的文件夹，其他文件夹都是Django框架自动生成的结构，用于网页演示。

* Com包存放公共使用变量和函数，其中macro.py中是宏变量名，utils.py中定义。
* Eval包存放的评价函数
* PreProc包存放预处理程序
* Single包存放多种相似度计算方法
  + cilin.py 同义词林方法
  + cwordnet.py 中文WordNet方法
  + hownet.py HowNet方法
  + ir.py 基于检索结果数的四种方法
  + linearcombination.py 线性结合方法
  + word2vec.py 词向量方法
* Counter包存放Counter-fitting方法，利用语义知识优化已知词向量。
  + counterFit\_new.py counter-fitting方法

核心算法如下：

|  |
| --- |
| **输入**：原始词向量空间*V*，反义约束集*A*，同义约束集*S*，相似约束集合*R*  **输出**：优化后新词向量空间*V'*  **L1：** *iter* = 0 and *V'* = *V* //初始化迭代次数，并将原始向量空间赋给新空间为初值  **L2： while** *iter* <= *max\_iter\_num* **do**  **L3：**  **for** each (*v*, *w*) in *A* **do** //遍历反义约束，计算AR项  **L4： if** *distance*(*V'* [*v*], *V'* [*w*]) < *delta* **then** //词向量间距离小于delta时  **L5：** *gradient = k*1 (*V'* [*v*] \* (*V'* [*v*] ·*V'* [*w*]) - *V'* [*w*]); //计算梯度值  **L6：** *gradient\_updates* [*v*] += *gradient*; *gradient\_count* [*v*]++;  **L7： end if**  **L8： end for**  **L9：**  **for** each (*v*, *w*) in *S* **do** //遍历同义约束，计算SA项  **L10： if** *distance*(*V'* [*v*], *V'* [*w*]) > *gamma* **then //**词向量间距离大于gamma时  **L11：** *gradient = k*2 (*V'* [*v*] \* (*V'* [*v*] ·*V'* [*w*]) - *V'* [*w*]); //计算梯度值  **L12：** *gradient\_updates* [*v*] -= *gradient*; *update\_count* [*v*]++;  **L13： end if**  **L14： end for**  **L15：**  **for** each (*v*, *w*) in *R* **do** //遍历相似约束，计算RSI项  **L16： if** *distance*( *V'* [*v*], *V'* [*w*] ) > *theta* **then //**词向量间距离大于theta 时  **L17：** *gradient = k*3 (*V'* [*v*] \* (*V'* [*v*] ·*V'* [*w*]) - *V'* [*w*]); //计算梯度值  **L18：** *gradient\_updates* [*v*] -= *gradient*; *update\_count* [*v*]++;  **L19： end if**  **L20： end for**  **L21：**  **for** each (*v*, *w*) in *all\_word\_pairs* **do** //遍历每对词，计算VSP项  **L22： if** *distance*(*V* [*v*], *V* [*w*]) < *distance*(*V'* [*v*], *V'* [*w*])**then**  **L23：** *gradient = k*4 (*V* [*v*] \* (*V* [*v*] ·*V'* [*w*]) - *V'* [*w*]); //计算梯度值  **L24：** *gradient\_updates*[*v*] += *gradient*; *update\_count* [*v*]++;  **L25： end if**  **L26： end for**  **L27：**  **for** each word in gradient\_updates **do** //更新梯度  **L28：** *V'* [*v*] += *gradient\_updates*[*v*] / *update\_count*[*v*];  **L29： end for**  **L30：** *iter*++;  **L31：end while** |

* + counterFit\_dev.py counter-fitting方法超参数调整
  + plotkkkp.py 画调参的图
* data存放数据的文件夹
  + corpus存放原始语料
    - NLPCC\_Formal500.txt

NLPCC评测测试数据，也就是PKU-500语料

* + - NLPCC\_Dry40.txt

NLPCC评测样例数据。

* + - MC30.txt

MC30翻译过来的中文数据，部分论文有用此数据（见朱新华2016）。

* + - SemEval50.txt

相近的评测SemEval2012的样例数据。

* + - train\_i.txt/test\_i.txt

根据PreProc/ split\_dev\_corpus.py切分得到的5组调参数据。

* + dicts存放字典文件
    - Cilin.dic及cilin.pkl 同义词林字典
    - fml\_org\_bdnews\_xieso\_w2v.bin 训练好的一个词向量模型
    - tongyis.txt/
  + pics存放画图文件
  + results存放结果文件
* WebDemo包，Django框架的主体部分
  + wordsim.py 功能实现的主体部分
    - search\_post(request)

接收POST请求数据，根据用户表单传来的【输入文本】和radio button选择的计算模式，调用不同的计算方法函数。

|  |  |
| --- | --- |
| 函数名 | 说明 |
| cilin\_webtest | 【同义词林】  输入用户想要评价的词对序列(w1\_list, w2\_list)，对于每一对词，调用Single/cilin.py中的sim2016()方法计算它们的相似度。  注意：第一次执行cs = CilinSimilarity()时会比较慢，因为加载同义词林耗时，所以利用pickle把同义词林加载后的对象保存（data/dicts/cilin.pkl），可以提升执行速度。 |
| hnet\_webtest | 【HowNet】 输入用户想要评价的词对序列(w1\_list, w2\_list)，对于每一对词，调用Single/hownet.py中的calc(w1, w2)方法计算相似度。 |
| wordnet\_webtest | 【中文WordNet】  输入用户想要评价的词对序列(w1\_list, w2\_list)，对于每一对词，调用Single/cwordnet.py中的cwn\_sim(w1, w2)方法计算相似度。 |
| word2vec\_webtest | 【word2vec】  利用gensim加载Word2vec预训练好的词向量模型，输入用户想要评价的词对序列(w1\_list, w2\_list)，对于每一对词用model.similarity(w1, w2)计算相似度。 |
| ir\_webtest | 【基于检索结果的相似度】  输入用户想要评价的词对序列(w1\_list, w2\_list)，调用Single/ir.py中的get\_nums\_list(w1l, w2l)方法分别得到检索结果数，最后调用ir.py中的四种方法得到jaccard，overlap，dice，pmi相似度。  jcd = web\_jaccard(num1, num2, num3)  ovl = web\_overlap(num1, num2, num3)  dice = web\_dice(num1, num2, num3)  pmi = web\_pmi(num1, num2, num3, N) |
| linear\_webtest | 【线性组合方法】  输入用户想要评价的词对序列(w1\_list, w2\_list)，调用上述方法得到不同的相似度，取其平均值作为最终相似度计算的结果。 |
| counter\_webtest | 【counterfitting方法】  首先要根据调参实验确定好合适的各个参数，然后调用Counter/counteFit\_new.py, 根据同义词典(data/dicts/tongyis.txt)和反义词典(data/dicts/merged.txt)，以及根据max(sim1, …, sim n) > 阈值得到相似对和相反对作为语义约束，优化一个已有的词向量，最后根据余弦距离定义相似度。 |

* + settings.py 配置文件
  + utls.py 路由配置文件
* static

Django框架中，存放css, js以及图片等静态文件。

* templates
  + WordSimilartiy.html

网站显示的html页面

* manage.py

Django框架自带，执行命令用

* StartServer.bat

脚本程序，双击即可打开服务器，然后便可在浏览器中输入访问网址，看到展示页面。

* 访问网址.txt

<http://127.0.0.1:8000/WordSim>



### 004 NLPCC会议

* NLP339

NLPCC会议论文latex源码文件，可直接编译得到PDF文件

* -437010\_1\_En\_69\_Chapter\_Author(原始).pdf 原始提交版本
* -437010\_1\_En\_69\_Chapter\_Author(checked&modified).pdf 修改版扫描件
* LNCS定稿问题.docx 修改说明

### 005 TALLIP期刊

* v1

acmsmall，NLPCC会议论文修改投TALLIP期刊第一稿Latex源码文件，大修

* v2
  + acmsmall，第二稿Latex源码文件
  + 修改说明

## 02 用户查询时间意图消歧

### 001 文献阅读

按照英文、中文、硕博论文三类整理的相关文献压缩包。

### 002 评测任务及结果

### 003 实验代码与数据

① 环境配置

代码使用Python实现，建议安装Anaconda（Python2.X）环境，缺少某个模块时请用pip install module\_name进行安装。

② 代码功能

* APIs

存放一些开源模块，每个模块的功能具体见注释。

* Comm

macro.py存放各部分公共用到的文件名等常量；utils存放各部分公共用到的通用方法。

* Evals

evaluationFunctions.py包括该任务的主要评价函数；calculateTFIDF.py计算TFIDF值；sentenceSim.py计算Query句子的相似度的一种简单实现。

* PreProcessing

预处理部分。

preModelProcess.py预处理入口程序，1、处理最原始的文件：格式化、标记时间表达式、斯坦福解析结果。产生文件：格式化的原始文件、heideltime和stanford的结果序列化文件、只有queries的文件；2、给新文件：格式化、标记时间表达式、斯坦福解析结果。

* Processing

extractFeatures.py，整个工程的主入口程序，首先，利用buildFeatureMatrix函数构建训练和测试语料的特征矩阵；然后，选择相应的特征并选择相应的分类器进行模型训练和测试。

sklearn\_classcification.py，所有用到的sklearn中的分类器。

feature\_selection.py，特征选择用。

* Tests

一些测试的小程序和画图程序，具体见每个代码注释。

* R

R语料代码，将谷歌趋势数据进行时-频域转换。

* corpus

所有用到的语料及中间文件。

|  |  |
| --- | --- |
| 名称 | 说明 |
| raw\_data | 该任务所有用到的【原始语料】 |
| TID\_format\_data | 该任务所有用到的统一【格式化语料】 |
| train\_test\_data | 几种原始语料组合而成的实验组语料 |
| doc（train\_test\_groups\_recoder.csv） | 实验分组说明 |
| GTData | 谷歌趋势数据-原始时域 |
| PLOT | 谷歌趋势数据-画图 |
| SPEC | 谷歌趋势数据-频域 |
| dicts | 放置词典 |
| feature\_matrixs | 为了方便分析，将特征矩阵存放到文件中 |
| only\_queries | 只有Query的语料文本 |
| pickled\_data | 为了加速实验，设置了pickle缓存的逻辑，中间pk文件存在这里 |
| translations | 英文翻译成中文的语料，为了扩充现有中文语料用。 |

### 004 NTCIR会议

### 005 已写未发专利

## 03 小论文汇总

### 001 已发表

1. Pei, J.\*, Zhang, C., Huang, D. & Ma, J. (2016). Combining Word Embedding and Semantic Lexicon for Chinese Word Similarity Computation. In International Conference on Computer Processing of Oriental Languages (pp. 766-777). Springer International Publishing.（EI检索会议，已发表）

2. Pei, J.\*, Huang, D., Ma, J., et al. (2016). DUT-NLP-CH @ NTCIR-12 Temporalia Temporal Intent Disambiguation Subtask, In Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, pp.253-257. Tokyo, Japan, Jun 7-10.（国际会议，已发表）

3. Ma, J., Pei, J.\*, Huang, D & Song, D. (2016). Syntactic Parsing of Clause Constituents for Statistical Machine Translation. International Journal of Computational Science and Engineering (EI检索期刊，录用待发表)

4. 马建军, 裴家欢\*, 黄德根. (2016). CRFs融合语义信息的英语功能名词短语识别. 中文信息学报, 30(6):59-66 （国内核心期刊，已发表）

5. 张聪, 裴家欢\*, et al. (2016). 基于语义图优化算法的中文微博观点摘要研究. 山东大学学报 (理学版) (国内核心期刊，录用待发表)

### 002 未发表

关于TID任务的，投稿EMNLP2016，未中。

* emnlp2016-tex，源码文件
* 图片
* 相关论文，以前发过的相关小论文
* Author Response Page for EMNLP 2016\_Conference on Empirical Methods in Natural Language Processing.pdf

返稿意见。

* 实验结果记录20160525.xlsx

# 3 基金

## 01 句子相似度计算2016

## 02 跨语言信息检索结项

# 4 其他

## 01 GPU平台搭建

### 001 GPU申请

GPU申请的一些文档。

### 002 电脑配置

电脑配置文档。

### 003 环境搭建

服务器具体的搭建方法。

## 02 规则修正分词工具

功能包括：按照规则合并分词结果，检查编码，过滤词性

具体见.bat脚本，双击可运行。