词汇相似度实验报告

语义计算与知识检索

韩 云

yunhan@pku.edu.cn

March 28, 2018

1 词汇相似度

词汇相似度计算是 MLP 领域最基本的任务，对于机器翻译、产品推荐、舆情控制、情感分析等众多应用将产生极大地促进作用。词语的语义相似度的计算方法主要有：

1. 通过语义词典，把有关词语的概念组织在一个树形的结构中来计算。
2. 基于语料统计的词汇语义计算，可以是对于语料库进行统计分析，也可以是将词语在隐含语义空间中用向量表示。
3. 基于 Web Search 的词汇予以计算，即通过返回的结果的数目之间的关系来完成词汇语义计算。

2 实验原理

本次实验分别从这三类方法进行展开，进行计算词汇相似度，以下是具体的算法分析。

1. 基于语义词典 (Wordnet) 的词汇语义计算
   - 基于路径的词汇相似度计算
     - 词在词典层次结构中的最短路径 Path-similarity

$$Sim_{path}(c_1, c_2) = -log\ pathlen(c_1, c_2)$$

     - Wu-Palmer 提出的最短路径 Wup-similarity, 其中 c_3 是 c_1 和 c_2 的最深公共子节点。

$$Sim_{wup}(c_1, c_2) = \frac{2 * pathlen(c_3, root)}{pathlen(c_1, c_2) + 2 * pathlen(c_3, root)}$$

     - Leacock Chodorow 最短路径加上类别信息 Lch-similarity

$$Sim_{lch}(c_1, c_2) = -ln\frac{pathlen(c_1, c_2) + 1}{2 * max\_depth}$$

   - 基于互信息的词汇相似度计算
     - Res-similarity

$$Sim_{res}(c_1, c_2) = -log\ P(LCS(c_1, c_2))$$

     - Lin-similarity

$$Sim_{lin}(c_1, c_2) = \frac{2 * log\ P(LCS(c_1, c_2))}{log\ P(c_1) + log\ P(c_2)}$$

     - Jcn-similarity

$$Sim_{jcn}(c_1, c_2) = \frac{1}{2 * log\ P(LCS(c_1, c_2)) - (log\ P(c_1) + log\ P(c_2))}$$

2. 基于语料 (Wikipedia) 统计的的词汇语义计算
   - 基于 word2vec 的词汇相似度计算

通过大量的语料获得词语之间的相互关系，然后把这种关系映射到隐含空间中，通过词语在空间中的向量的距离来计算词汇之间的相似度。这里首先获取基于 word2vec 项目给出的 text8 作为语料，然后调用 gensim 的 Word2vec 的模型进行训练得到词向量，随后利用词汇向量模型计算词语之间的相似度。

3. 基于检索 (Web Search) 页面数量的词汇相似度计算
   根据 Web Search 的结果，$H(word)$ 表示搜索之后返回的结果的数目，具体的各个算法如下：

   - Jaccard_score

$$Jaccard\_score(c_1,c_2)=\begin{cases}0 & if H(P\bigcap Q)\leq 5\\ \frac{H(P\bigcap Q)}{H(P)+H(Q)-H(P\bigcap Q)} & otherwise\end{cases}$$

   - Overlap_score

$$Overlap\_score(c_1,c_2)=\begin{cases}0 & if H(P\bigcap Q)\leq 5\\ \frac{H(P\bigcap Q)}{\min\{H(P),H(Q)\}} & otherwise\end{cases}$$

   - Dice_score

$$Dice\_score(c_1,c_2)=\begin{cases}0 & if H(P\bigcap Q)\leq 5\\ \frac{2*H(P\bigcap Q)}{H(P)+H(Q)} & otherwise\end{cases}$$

   - Pmi_score

$$Jaccard\_scor(c_1,c_2)=\begin{cases}0 & if H(P\bigcap Q)\leq 5\\ log\frac{H(P\bigcap Q)*Googlepages}{H(P)+H(Q)} & otherwise\end{cases}$$

3 实验数据及环境

- 实验数据
  MTURK-771。
- 实验环境 (Python3.6)
  - Nltk 工具包中的 Wordnet, Wordnet_ic, gensim(Word2Vec)。
  - 爬虫相关的库: urllib, BeautifulSoup。
  - 语料：Wikipedia enwiki8 的 text8。
- 评价方法
  斯皮尔曼相关系数为标准。

4 实验结果和分析

4.1 实验结果

根据每个算法计算的词汇相似度的结果，计算与实际人工调查的结果的斯皮尔曼相关系数如下：

表 1: 词汇相似度算法斯皮尔曼相关系数结果

词汇相似度方法	斯皮尔曼相关系数
wordnet_path	0.4984890974327451
wordnet_wup	0.45500457247354575
wordnet_res	0.4960385026198078
wordnet_path	0.34968140162086736
wordnet_lin	0.48121176224168344
wordnet_jcn	0.4749514700437678
corpus_word2vec	0.5065981534726528
web_search_jaccard	0.033354730736292915
web_search_overlap	-0.01740101202231092
web_search_dice	0.033354730736292915
web_search_pmi	-0.03302725663731394

根据结果绘制的斯皮尔曼相关系数条形图如下：

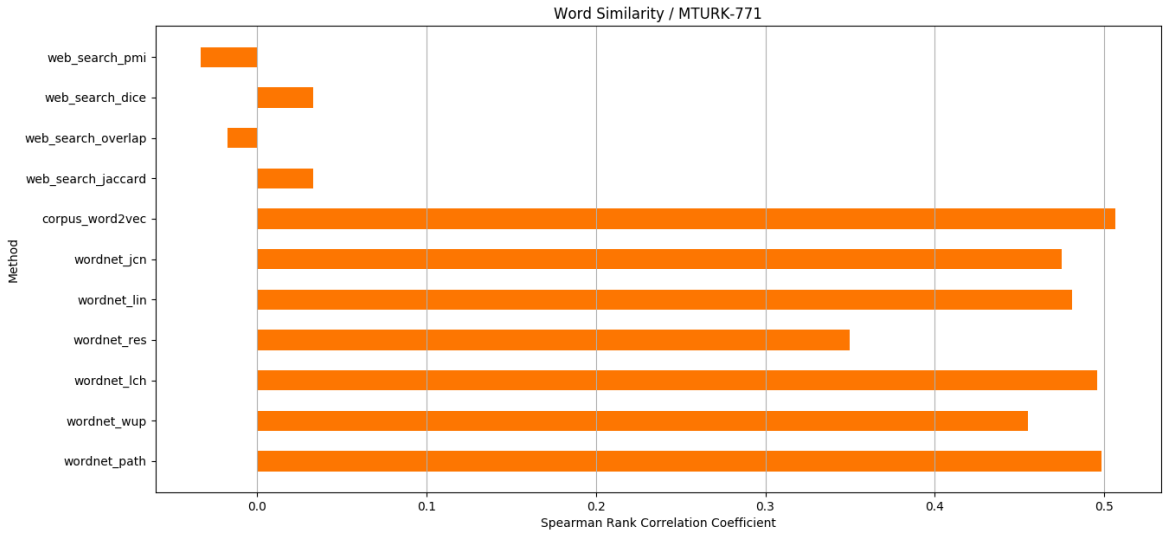

图 1: 词汇相似度方法 & 斯皮尔曼相关系数

4.2 结果分析

从总体来看，利用 Wikipedia 语料的方法好于基于 WordNet 和 PageCount 的方法。

由于 WordNet 的信息量比较有限，一些词语没有被收录到语义词典中，而且收录的词语不同词性之间也无法计算语义相似度。PageCount 的则只考虑了页面搜索数量，因此相关系数也较低。但是基于 WordNet 的方法好于 PageCount 方法，因为 PageCount 方法没有考虑到词语之间的词汇层级关系和语义关系，而且 PageCount 中基于 PMI 和 OverLap 的计算方式的斯皮尔曼相关系数为负值，表示此种方式并不适用于这个测试集。

考虑到计算复杂性问题，Word2vec 部分 Wikipedia 语料作为训练数据（text8 大小约为 100MB），导致实验的结果没有那么明显，如果使用更多语料，可能会获得更好的结果。

通过以上的实验结果，我们可以近似得出如下分析结果：

1. 基于语义词典的词汇相似度计算，当且仅当两个词语之间存在一条路径时才能进行词汇相似度计算。计算直观而且简单，可以计算出表面上不相似的词语之间的相似性，但是受人的主观影响比较大；对于不被语义词典包含的新词、不同词性的词语不能计算相似度，而且大部分方法依赖于上下位层次关系，对于形容词和动词并不完善。

2. 基于语料库 (基于 Web-Search 也可以看做语料库的一种。) 的词汇相似度计算，当且仅当两个词语处于相似的上下文之间时才能进行计算。根据词语的形态、句法、语义等特点计算，更为客观，但是依赖选取的语料库的优劣，比较容易受噪声的影响。

5 参考文献

Large-scale learning of word relatedness with constraints. KDD 2012: 1406-1414.