具体试错和解题思路

Sub-task2 论文的冷启动消歧

第一名 ECNU_AIDA

队员：汤旭东， 指导老师：张伟

华东师范大学 计算机科学与技术系

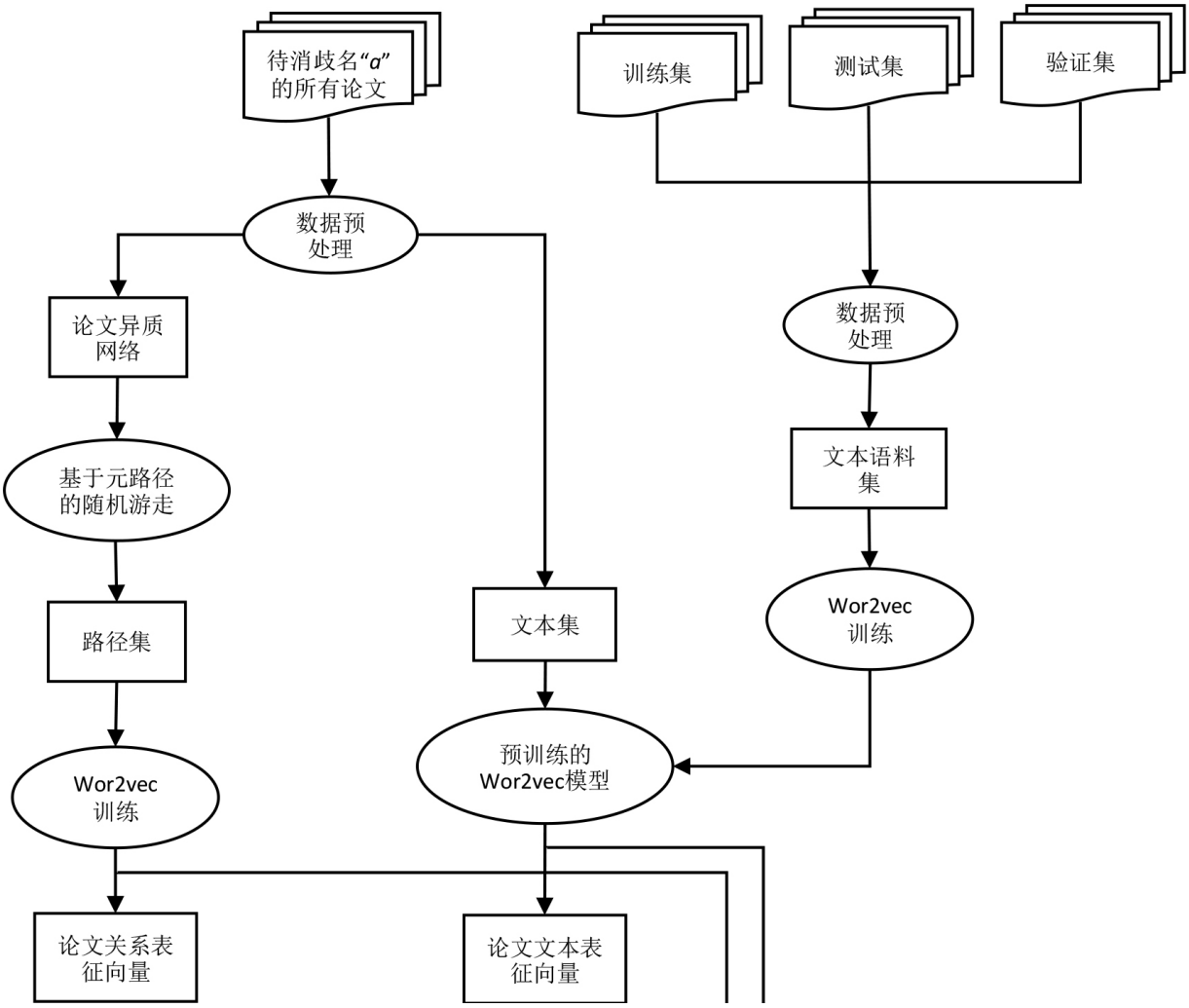
摘要

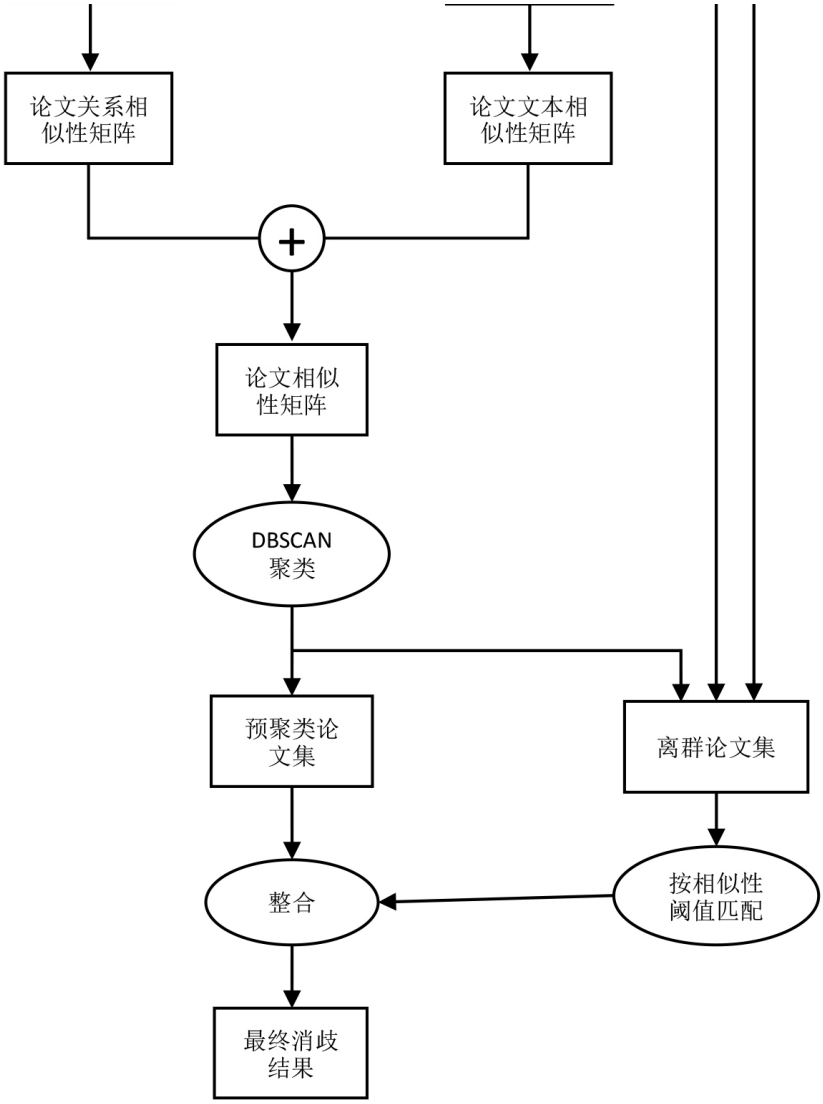
主要参考第一届WhoIsWho比赛(https://www.biendata.xyz/competition/aminer2019/)的第一名的分享: **基于网络嵌入和语义表征的作者名消歧**(https://www.biendata.xyz/models/detail/3968/)，并尝试从1.网络边的建立方式和2.语义表征方法两个方面对原方案进行优化。

1.整体思路

对于某个需要消歧的名字，根据其名下所有待消岐论文之间的网络关系信息和文本信息求出论文表征向量，进而求出论文之间的两两相似度，得到论文相似性矩阵，使用聚类算法和基于相似度阈值的匹配方法将论文划分为不同的簇，则每个簇代表一个特定作者的论文集。

整体框架如下：





2.特征分析

把该问题看作是对一个论文集的聚类任务，且不指定聚类簇个数，即K值。

首先，分析每篇论文的特征，论文的特征包含title, abstract, author, venue, organization, year, keyword字段. 把这些特征划分为两种类型，一种是语义特征，一种是离散特征。

语义特征指的是可以具有语义信息的文本特征，例如title, abstract, keyword，这些文本可以使用语义表征学习模型如word2vec等转化为文本语义向量。在后续的实验中，我们认为venue, organization, year也具有弱语义信息。

离散特征指的是本身的文本信息没有很大的价值，例如author，作者名称本身的语义并没有作用，一个作者只有在两篇文章中同时出现时才有作用，这表示两篇论文有一个共同作者，则它们之间的相似性较大。因此我们把这类特征成为离散特征，只能用来转换为论文间的关系。在后续的实验中，我们发现organization也属于离散特征，因为其中的有些词如地名可以用来搭建论文之间的关系。

具体实施上，我们定义author, organization为离散特征，定义title, venue, organization, year, keyword为语义特征。

基于对以上两种特征的认识，我们的思路是使用语义特征学习论文的文本表征向量，利用离散特征来构建论文之间的关系，如有共同作者关系，有机构相似性关系，以此构建论文网络，通过网络嵌入学习方法来学习论文的关系表征向量。再使用聚类算法对论文进行聚类。

3.论文表征学习

这部分我们介绍如何学习论文的两种表征向量。

3.1 论文关系表征学习

此部分的作用是学习到每个论文的关系表征向量。可以看作是先搭建论文异质网络，然后利用网络嵌入的方法学习到每个论文节点的表示向量。本部分用到的特征有**co-author**和**co-organization word**.

网络嵌入（Network Embedding）模型尤其是异质网络嵌入模型已经有了很多研究成果，我们使用的网络嵌入模型主要参考DeepWalk[1]和Metapath2vec[2]。

3.1.1 异质网络构建

对于每一个需要消歧的名字，将其对应的所有的论文之间的关系抽取出来，构建出一个论文异质网络，如图1所示。这个异质网络包含一种类型的节点：论文（每篇论文代表一个节点），两种类型的边：CoAuthor，CoOrg。

CoAuthor代表两个论文之间有共同作者（不包含需要消歧的名字），边上的度代表拥有共同作者的个数。例如如果两篇论文之间有共同作者，那么就在它们之间构建一条关系名为CoAuthor 的边，同时这条边具有共同作者数目的属性，如果有1个共同作者，这个关系的权重就为1，如果有2个共同作者，那么权重就为2，以此类推。

CoOrg代表两个论文中待消歧名字的机构的相似性关系，边上的度代表两个机构共词的数量。例如，两个论文的机构有相同出现的词，且这个词不是停用词，那么就在它们之间构建一条CoOrg的边，这边相应的也有数目的权重，如果有一个共现词，那么权重值为1，如果有两个共现词，那么权重为2，以此类推。要注意的是我们**只使用了待消歧名字的机构信息**。

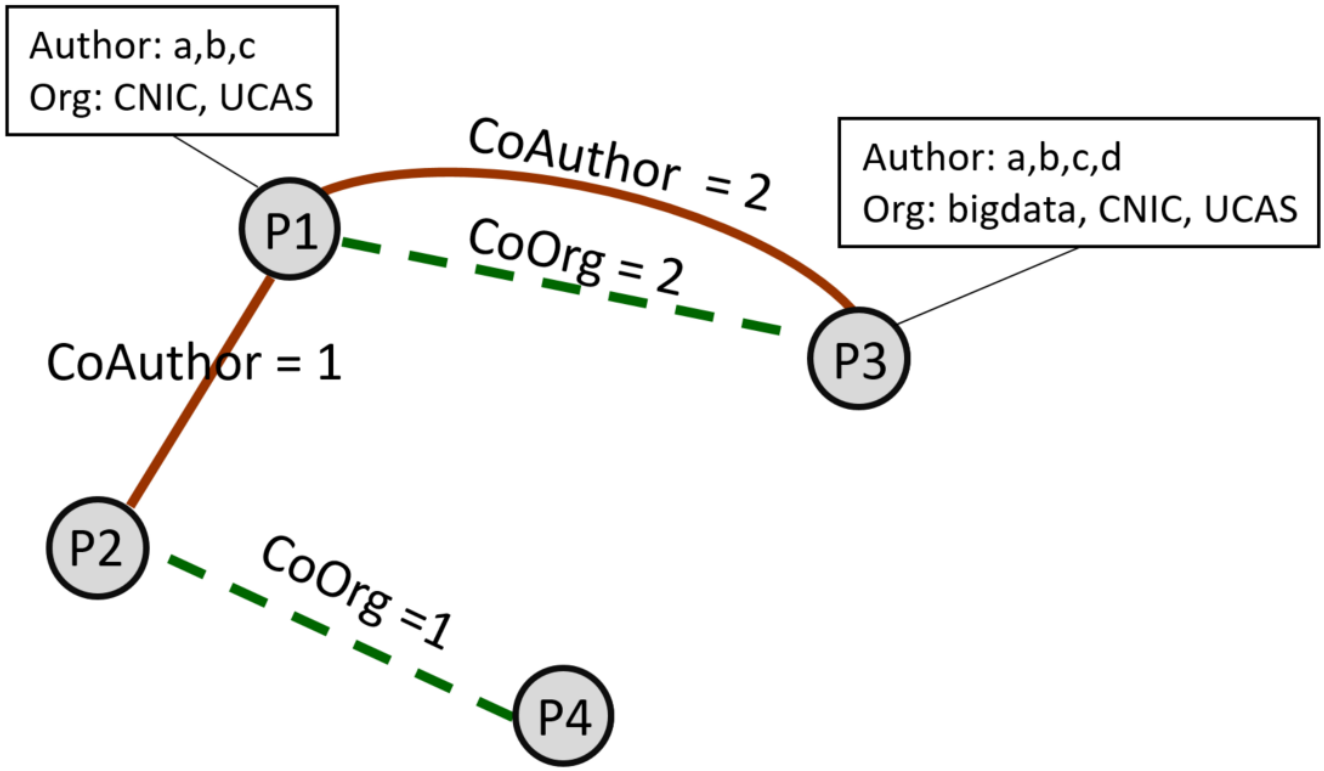


图1. 论文异质网络示意图。其中a是待消歧作者名，Org代表a所属的机构。

3.1.2 基于元路径的随机游走

构建完上述论文异质网络，使用基于元路径的随机游走生成由论文id组成路径，把这些路径作为word2vec的输入，最后得到每个论文id对应的关系表征向量。

具体来说，在论文异质网络中轮流选择每一节点，以该节点为初始节点，在节点之间的边进行随机游走，把游走的路径保存下来作为word2vec的训练语料库[1]。基于元路径的随机游走的意思是在边上随机游走时，并不是完全随机的，而是由元路径指导的[2]。

规定在随机游走的过程中，按照p1->CoAuthor->p2->CoOrg->p3这样的元路径顺序进行游走，每一次游走过程中，按照当前元路径规定的某种类型的边选择下一个节点，随机选取一个通过该类型边与当前节点相连的节点作为下一个节点，在每一条长路径中，重复采样若干次这样的元路径，即前一条元路径的最后一个节点作为下一条元路径的第一个节点。最终迭代直到达到一定的次数（代码中的定义为walklenth），则另选一个节点作为开始节点进行游走，最终生成若干这样的长路径，每个路径点是论文的id，并将每个长路径按行存储，生成训练语料库。

同时，在某个节点在元路径指导下朝着某类型的边随机选择下一个节点游走的过程中，会考虑到边的权值，权值越大，说明两个节点的关系越密切，00那么节点沿着这条关系跳转的概率就越大，我们规定该概率与权重是正比例关系。例如，上图中，若p1为当前节点，下一跳的关系是CoAuthor，那么与p1有该关系的两个节点分别是p2和p3，根据它们之间该边的权值，那么从p1游走到p2的概率是1/3，游走到p3的概率是2/3。

在某些情况下，有些关系对于一些论文来说是缺失的，例如某个论文中所有的作者并没有出现在其他任意一个论文的作者列表中，那么对它来说CoAuthor这个关系是缺少的，当出现这种情况时，就采用更灵活的策略，即根据元路径中当前缺失关系的下一个关系游走，对于上面说的那篇论文来说，就转而根据它的CoOrg关系进行游走。

这样，每一轮选择图中的每个节点为起始点开始随机游走，我们在代码中定义轮数为numwalks，这样就生成了numwalks*N条路径组成路径集，其中N是节点数量。实际中我们设置numwalks为5，walklenth为20。

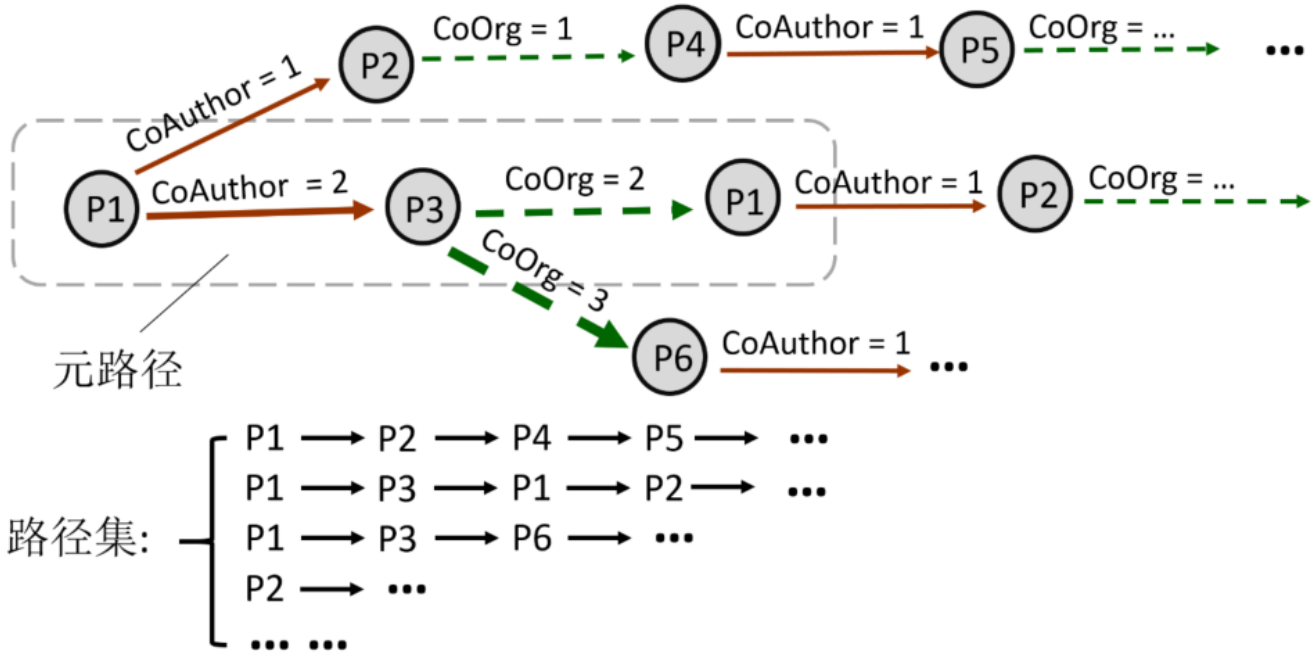


图2. 基于元路径的随机游走生成路径集。箭头的粗细代表游走到下一节点的概率。

注：在实际上从效率上考虑，我们没有真的生成一个异质图，而是只存储了paperid: author，author: paperid, paperid: orgword, orgword: paperid这样的键值对，其中orgword代表机构名称里的一个词。在游走经过两个论文节点一条CoAuthor边时，先用paperid随机找到其中一个author，再用这个author去找到下一个paper，同理，在游走经过两个论文节点一条CoOrg边时，先用paperid随机找到其中一个词，再用这个词去找到下一个paper。

这样随机游走的方式和上述策略是等价的，但不用构建完整的异质图，更加节省时间。此部分代码详见utils.py中的MetaPathGenerator类。

3.1.3 使用word2vec得到节点关系向量

我们使用gensim库中的word2vec的skip-gram模型训练上节的路径集，将路径集看作语料库，语料库中的论文id看作词汇，最后得到每个论文id对应的表征向量。由于这个语料库中实际保存的是节点之间共同作者和机构名称相似性的关系，所以我们把这个表征向量叫做节点关系表征向量。Word2vec训练参数如下：

```
mpg.generate_WMRW("gen_rw/{self.mode}/RW.txt",5,20)   #生成路径集
sentences = word2vec.Text8Corpus(r'gene/RW.txt')
model = word2vec.Word2Vec(sentences, size=100,negative =25, min_count=1, window=10)
```

在采样的路径中，我们并不保存起始点，这样对于图中的孤立节点（即没有边与其他节点相连的节点），这个节点对应的论文id不存在于路径集中，也无法得到其节点关系表征向量。我们把这些节点的关系表征向量设为全0向量，并保存到离群论文集中，后续再做处理。

在训练word2vec的时候，我们采用了Bagging的思想，采用上节中的策略采用k个路径集，使用k个路径集训练得到k个word2vec的模型，生成k组论文向量，每组论文向量求余弦相似性矩阵，再对这k个相似性矩阵求均值，得到最终的论文关系相似性矩阵。

3.1.4 试错过程

我们只使用了co-author和co-organization word两个特征，其实venue特征和organization是有一定的相似度的，都是一段短文本，可以搭建节点之间的venue相似性关系的边，我们也试过把venue特征加入异质图，采用相似的元路径随机游走策略学习向量，但效果并不好。我们认为的原因是，由于我们相当于使用词共现次数来确定两个论文organization的相似度的。venue的文本中有许多带有学科属性的词，这些词的共现较为常见，有共词对证明两篇论文是同一个作者所写的贡献不高，因此会带来一定的噪声。而organization中多为地名，学校名，这些词的共现相对更有说服力。

除了待消歧作者名的organization，每篇论文的其他作者也包含organization，但我们认为这些organization的用处不大，并不是强特征，经过实验也验证了这一点。

Co-organization word是重要的特征，但找待消岐author的organization字段并不是很直接，因为相同名字的写法不止一种，例如图3所示待消岐名字"Wei Chen"在数据中的不同写法；还有比如中文的名字，可以姓在前也可以名在前(如"Bo Li" -> "Li B" or "B Li")，姓和名都可能缩写到只保留首字母(如 "Liang Gao" -> "L. Gao" or "l gao")；还有，数据中还有中文名直接写以汉字保存的情况，处理时把他们转成了拼音写法；英文名同样有缩写的情况，而且可以带middle name也可能不带。除此外还有一些原名不是英文、中文的名字，来源可能是日文韩文法文等，相同名字可以有不同的英文写法，比如来自日语的"tadashi" 写作 "takashi" 也是可以的(这部分通过看case，哪个名字待消岐的作者找不到的情况比较多就看看)。姓、名；中文名字的间隔，可能是空格可能没有，可能是"-"; 这部分的代码详见utils.py中的 match_name函数，精细处理后训练集/验证集的最终F1得分提升1%-1.5%

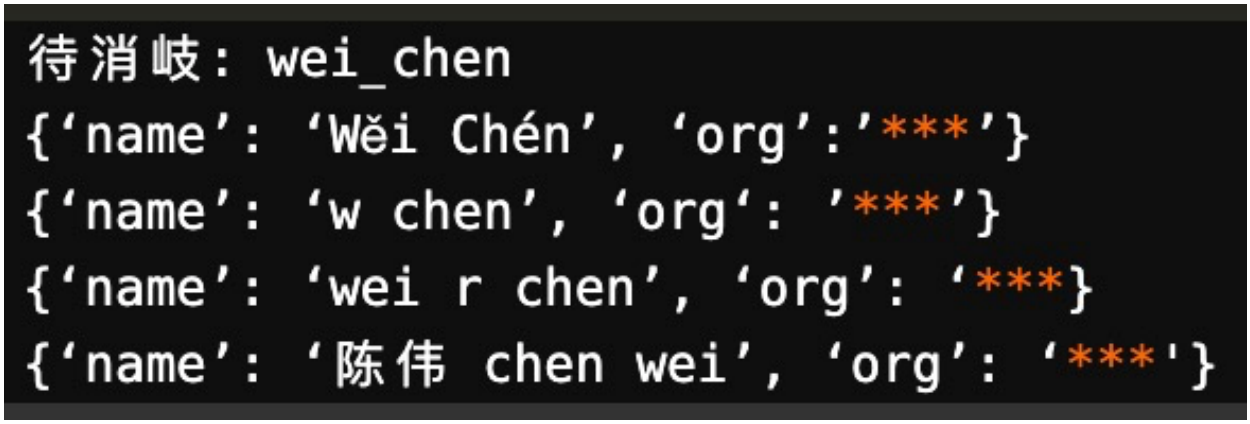


图3.待消岐名Wei Chen在数据中的不同写法

Co-organization word是比较粗粒度的，比如两篇文章待消岐作者的organization都带有"BeiJing"这样的地名；或"Medical"这样表示专业的词；如果以此为据把两篇文章建立联系是很粗犷的，但这些词确实又提供了一定的线索，如果把他们纳入停用词，模型效果会下降；尝试两篇文章只有organizaiton完全相同或按编辑距离等模糊匹配方式的相似度高于一定阈值才有边相连，模型效果还是下降了，猜测因为这样带来的稀疏性的问题更大，超过了收益。这个问题有点像分词粒度的问题，或许找到一个中间的粒度才能有更好的效果，因为时间原因没有再尝试了。

3.2 论文语义表征学习

此部分的作用是生成论文语义表征向量。

首先将训练集，验证集和测试集中的文本提取到一个文本语料集中，具体来说，对于每一篇论文，我们提取了其每个作者的organization，论文的title，论文的abstract，论文的venue这些文本，并按行存储到一个txt文件中。在测试集未出时，我们只使用了训练集和验证集的文本。

接着我们以这个txt文件为语料集，使用gensim的word2vec模型训练词向量。此部分的代码在“train word2vec” cell模块中，Word2vec训练参数如下：

```
sentences = word2vec.Text8Corpus(r'extract_texts/train_valid_test.txt')
model = word2vec.Word2Vec(sentences, size=100,negative =5, min_count=2, window=5)
```

对于每一篇论文，我们生成它对应的语义表征向量过程如下：将该论文的title, venue, organization, year, keywords 以空格间隔，合成同一段文本，对这个文本进行预处理，首先把字母小写化，去除各种非字母的符号，接着去掉多余的空格，以空格分词，去掉停用词和长度小于3的词。将这些词都通过上段训练好的word2vec模型生成词向量并求均值，最终得到论文的语义表征向量。

对每个待消歧的名字，得到其所有论文的语义表征向量，当有的论文的所有词都不存在于word2vec模型中时，将其语义表征向量置为全0，并保存到离群论文集中，后续再做处理。最后求得论文两两间的余弦相似度，得到论文语义相似性矩阵。

3.2.1 试错过程

最开始根据经验，只用了title，keyword和abstract两个文本信息来生成论文语义相似性，后来发现abstract的语义效果并不好，原因是其中的噪声词太多，实际上只用使用几个主题词和关键词的语义信息就可以代表该论文的语义特征了，因为这些词代表了论文的主题，而title和keyword正是富含这些词，因此，我们只在词向量训练时使用了abstract。

另一方面，发现文本序列信息的用处也不大，因为基本上我们使用的都是短文本。

把organization和venue加入文本信息中，发现效果有提升，我们认为词向量能够很好的量化这两个特征的文本相似性。

把year也加入文本，这些年份数字也分布在文本集中，因此也可以生成词向量，而且相近年份的词向量具有更高的相似性，比如与2012最相似的年份词是2011和2013。这量化了两个论文的发行年份的相似性。

考虑用更多的语料训练word2vec，用了往届比赛的数据，同样的方式提取文本；训练集上效果没什么变化，验证集上效果下降了；可能是数据的分布不一致。

考虑到abstract, organization这样不同的字段，词的分布会很不同，尝试分别训练word2vec模型后用向量拼接的方法得到文本表示；但因为各个待消岐名字下organization，abstract等缺失情况各不相同，各自拼接的维度数也不好指定，最终效果还不如上面的融合各个字段的文本的word2vec模型

考虑更精细的文本处理，去掉更多停用词，数字等；更能提现关键词的信息但会丢掉一些信息；效果略有下降，可能是数字比如年份或者一些药物的关键词带有数字或者特殊的写法，能体现一些文本的特征；文本处理太精细反而丢失这些特征。

考虑各个词的权重，使用了TF-IDF权重，试过全部paper的全局TF-IDF和每个待消岐名字下文章自己计算Local TF-IDF，训练集上效果有一定程度的提升，但验证集上效果下降了，所以最终也没有在测试集采用。

4.论文划分

这部分我们根据上节得到的两个论文相似性把论文划分到不同的簇中，每个簇代表这些论文为同一个作者所写。

4.1 预聚类

第三节中，我们得到了论文关系相似性矩阵和论文语义相似性矩阵，首先我们把这两个矩阵相加求均值。得到最终的论文相似性矩阵。我们考虑过加权相加，以测试哪个相似性矩阵具有更高的比重，最终根据训练集的结果得出这两个矩阵的权值比应设置为1:1。

根据这个论文相似性矩阵，将这个矩阵输入DBSCAN中，得到聚类结果。

```
pre = DBSCAN(eps = 0.2, min_samples = 4,metric ="precomputed").fit_predict(dis)
```

我们把DBSCAN的参数min_samples设置为4，这意味着一个簇中最少的论文个数为4，这会产生一部分已经划分好的论文簇和许多label为-1的离群点，这些离群点不属于任何簇。我们把这些label为-1的论文加入第三节提到过的离群论文集中。除了这些在离群论文集中的论文，我们把其他论文的聚类结果作为这些论文最终的聚类结果。

4.2 匹配离群论文

我们目前得到了一些已经划分好的簇，里边包含着已经聚类好的论文，我们把这些论文合起来的集合叫做预聚类论文集，还有一个离群论文集，且这两个集合不相交。这一步的操作是将这些离群论文集里的论文用阈值匹配的方法重新分配给已经聚类好的簇或者新的簇中。具体操作如下：

首先对于离群论文集中的每一篇论文，比较它与每个预聚类论文集中的论文，得到跟它匹配相似度最高的一个论文，如果他们两个的相似度不小于阈值α，则把前者分配到后者所在的簇中；否则把它单独归为新的一个簇。

做完上步，对于离群论文集中的每一篇论文，再比较它与离群论文集中其他每个论文匹配相似度，如果两者的相似度不小于阈值α，则把后者分配到前者所在的簇中；否则不变。

其中两篇论文pi和pj的匹配相似度s(pi,pj)的定义如下：

```
s(pi,pj) = （pi 和pj的共同作者数）* 1.5
+ tanimoto（pi的venue，pj的venue）
+ tanimoto（pi中待消歧名的organization，pj中待消歧名的organization）
+ (pi和pj中title的共词数）* 0.33
```

其中tanimoto(p,q)指两个序列的tanimoto相似度，定义如下：

```
def tanimoto(p,q):
    c = [v for v in p if v in q]
    return float(len(c) / (len(p) + len(q) - len(c)))
```

阈值α我们取为1.5。

生成和匹配离群论文的原因是这部分论文往往特征不够明显，造成其与其他论文之间的相似度比较小，或者这些论文本身就是属于一个论文数较少的作者，这些形式的论文用我们上述所说的表征向量学习方法的效果不如使用直接使用特征进行匹配的方法的效果好。

经过上述操作，将离群论文集中的论文也分配到了各自的簇中，将离散论文集和预聚类论文集整合，这样就得到了所有论文的聚类结果，这个聚类结果就是我们最终对于某待消歧名字的消歧结果。

其他

我们对所有上文未提到的文本预处理的操作都是一样的，首先把字母小写化，去除各种非字母的符号，接着去掉多余的空格，若文本需要分词，则在分词后去掉停用词，和长度小于2的词。使用的停用词表和符号表如下：

```
r = '[!“”"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~—～’]+'
self.stopwords = ['at', 'based', 'in', 'of', 'for', 'on', 'and', 'to', 'an', 'using',
'with', 'the', 'by', 'we', 'be', 'is', 'are', 'can']

self.stopwords_extend = ['university', 'univ', 'china', 'department', 'dept',
'laboratory', 'lab', 'school', 'al', 'et', 'institute', 'inst', 'college', 'chinese',
'beijing', 'journal', 'science', 'international', 'key', 'sciences', 'research',
'academy', 'state', 'center']

self.stopwords_check = ['a', 'was', 'were', 'that', '2', 'key', '1', 'technology', '0',
'sciences', 'as', 'from', 'r', '3', 'academy', 'this', 'nanjing', 'shanghai', 'state',
's', 'research', 'p', 'results', 'peoples', '4', 'which', '5', 'high', 'materials',
'study', 'control', 'method', 'group', 'c', 'between', 'or', 'it', 'than', 'analysis',
'system',  'sci', 'two', '6', 'has', 'h', 'after', 'different', 'n', 'national',
'japan', 'have', 'cell', 'time', 'zhejiang', 'used', 'data', 'these']
```

参考文献

[1] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
[2] Dong, Yuxiao, Nitesh V. Chawla, and Ananthram Swami. "metapath2vec: Scalable representation learning for heterogeneous networks." Proceedings of the 23rd ACM SIGKDD international conference on knowledge

discovery and data mining. ACM, 2017.