现在自然语言处理（NLP）、机器学习（ML）等领域可谓是火爆异常，而随着社交网络的爆发式发展，自然也有很多研究者对社交媒体上的文本分析产生了兴趣。而说到社交媒体这块，玩的最溜、影响最大的可谓是现任美国总统特朗普（Trump）的推特（twitter）了，这哥们不仅常常一天狂发数条推特，还经常在上面直接宣布一些重大决定，或者耿直直接对政治对手进行攻击，人称“推特治国”。因此。用一些机器学习方面的算法来分析他的推特会不会产生一些有趣的结果呢？考虑到此， 这个练习项目就这么开始了。

数据获取

项目的第一步当然是下载足够的特朗普tweet数据。twitter提供了官方的API供用户用python语言与客户端交互；研究者只要提供自己的相应密钥信息，就可以通过它直接下载twitter相关的数据，具体方法可以参见相关介绍文档[1]。但是，官方API有一个很大的限制，就是下载的时候只能回溯同一个用户之前的3200条左右的tweet，再之前的就不给了。。。这让人有点失望，说好的大数据就这么没了。由于官方API的使用不是很方便，因此实际上我调用的是更上层的tweepy[2]，再通过它收集@readDonaldTrump这个账号的相关数据，包括发表过的推特的文本、发布时间、发布语言、点赞数、转发数、发布平台等。

下载程序参考[3]的架构，包括两部分功能：第一个部分负责下载并保存离当时最近的3000条tweet数据；第二个部分可以在稍晚时候运行，负责下载特朗普在之后新发的tweet。自从参加竞选以来，特朗普已经化身为发推狂，每天都要连发数条，所以在这个项目期间又极大的丰富了数据数量。最后用于分析的tweet从2016年6月2日5点开始，到2017年9月14日9点结束，共3685条tweet，保存为dataframe格式，如下图：

id	text	favorite_count	retweet_count	lang	source	created_y	created_m	created_d	created_h	created_min
908161523180285000	The "deplorables" came back to haunt Hillary.They expressed their feelings loud and clear. She spent bi	43818	10732	en	Twitter for iPhone	2017	9	14	2	52
908160218995068000	Crooked Hillary Clinton blames everybody (and every thing) but herself for her election loss. She lost th	56288	14424	en	Twitter for iPhone	2017	9	14	2	47
908154067658174000	China has a business tax rate of 15%. We should do everything possible to match them in order to win v	40420	10064	en	Twitter for iPhone	2017	9	14	2	22
907946177022369000	With Irma and Harvey devastation, Tax Cuts and Tax Reform is needed more than ever before. Go Cong	81288	18023	en	Twitter for iPhone	2017	9	13	12	36
907930425657626000	I will be traveling to Florida tomorrow to meet with our great Coast Guard, FEMA and many of the brav	77913	13846	en	Twitter for iPhone	2017	9	13	11	34
907928888587808000	The approval process for the biggest Tax Cut & Tax Reform package in the history of our country w	70703	16296	en	Twitter for iPhone	2017	9	13	11	28
907698529606541000	It was a great honor to welcome Prime Minister Najib Abdul Razak of Malaysia and his distinguished del	46839	13097	en	Twitter for iPhone	2017	9	12	20	12
907675638055743000	Congratulations to Eric & Lara on the birth of their son, Eric "Luke" Trump this morning! https://t.c	104106	16077	en	Twitter for iPhone	2017	9	12	18	41
907592460070768000	The devastation left by Hurricane Irma was far greater, at least in certain locations,than anyone thought	80555	14227	en	Twitter for iPhone	2017	9	12	13	11

数据清洗

接下来我们需要对“text”这一列进行处理，也就是每条tweet的文本部分，其编码为utf-8。对于一般意义上的NLP分析，有很多语义分析工具可以使用；

但是，twitter文本具有一些特殊的语法，因此这里还是需要做一些预处理工作。

首先假设文本中的每个单词都是用空格隔开的（事实上这也是英文的一般情况，不像中文还需要做分词工作）；这样可以用自然语言处理包NLTK[4]中的whitespaceTokenizer把tweet中每个用空格隔开的文本单元分成初步的token，再将这些token的字符全部转为小写。之后按照下列次序辨别一些特殊token：

7. 如果开头为“https”，这个token辨识为网站链接（link）
8. 如果开头为“RT”，则该token为转发的tweet（retweet）的固定开头
9. 如果开头为“@”，则为推特的用户名
10. 如果开头为“#”，则代表“hashtag”（twitter中用来标注线索主题的标签）
11. 如果开头为“&”，则基本上是tweet中的“&”符号，在utf-8的文本中以“&”代表（目前暂未发现反例）
12. 通过一个正则表达式，辨识token是否为email地址

排除掉这些特殊token之后，对于剩余token再用Tokenizer再做一次分词（我用的WordPunctTokenizer），这样可以拆分那些没有被空格隔开的、被标点符号、连字符连接的多词token。最后对每个token再做一次检查，如果是全部由拉丁字母和数字组成的，那基本可以认为是普通类型的token（标记成normal）；否则则说明它包含一些特殊字符，例如原本tweet里的表情符号被强制转换成utf-8文本之后，会变成一些乱码；或者是一些外国语言的人名之类的，这些token统一标记成“special”类型。在后续的NLP分析中，只考虑normal类型的token。

这样，每个tweet就转化为了一个token组成的list，每个token有自己对应的类型。同时，我们还对tweet的整体做了一些识别；比如，tweet的首个token是“RT”的话，则第二个token自动对应转发来源，且整个tweet都标记成“retweet”，这样后续的一些分析可能就不再考虑这种转发的推特（例如，retweet的点赞数一般会明显小于原创的tweet）；可以把在单/双引号内部的

token全部标记成“被引用状态”，后面可以决定是否考虑这些内容（因为引用的可能是别人的话；但其实也很难说，因为也可能只是表示讽刺之类的）；还可以根据token辨识的结果生成一个不含特殊字符的“干净版”的文本，然后用NLTK的sent_tokenize函数进行分句，从而可以进行每个sentence的分析等等。

这里我们随机挑选一个tweet，检验一下我们的文本分析的结果，例如特朗普在今年6月22日发的一条tweet：

注意下方插入的视频，在转化为utf-8文本数据的时候，它会变成一个链接。我们的分析结果为（括号中为辨识到的token类型）：

we(normal) '(special) re(normal) all(normal) thinking(normal) of(normal) you(normal) stevescalise(tweetUser) teamscalise(Hashtag) https://t.co/yqf6exhm7x(link)

初步统计

数据清洗完成之后，我们可以进行一些基本的分析了，比如建立一个简单的词袋（bag of words，BOW）模型。这里我们只统计原创的（非转发）tweet中类型为normal的tokens，并且过滤掉停止词（stopwords，如没有实际语义的to、for、a等）、单纯的数字和部分（无意义）的地名等。统计得到数据中共有57589个单词、31503个非停止词、5206个独特的单词。使用频率最高的几个单词为great（516次）、thank（397次）、hillary（362次）、clinton（261次）、people（241次）、trump（209次）、america（198次）等，很符合我们对trump用语的一贯印象。最后将词袋里的单词画成词云（word cloud），作为一个形象化的展示：

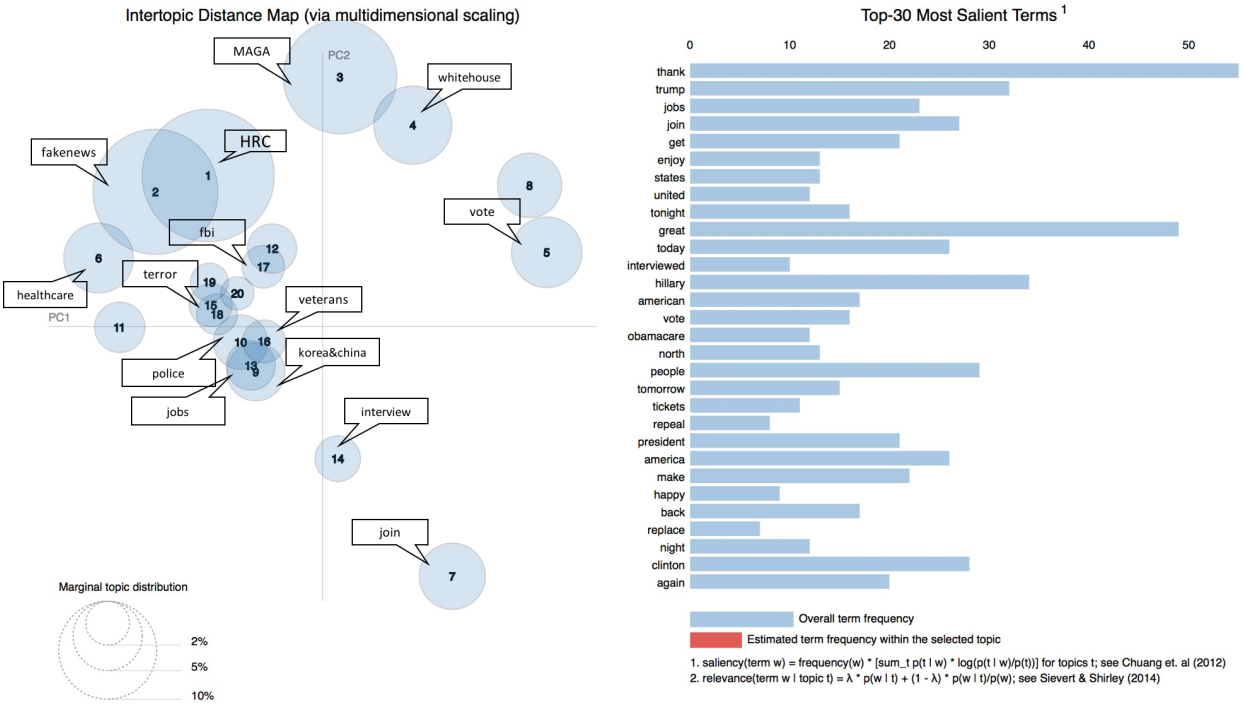
无监督学习

主题分析是NLP的一个重要组成部分，在这里也是一个很有意思的研究方向：特朗普的每条tweet可能分属不同的主题，比如竞选拉票、喷政治对手、喷外国、讨论工作问题、讨论医保问题等等。因为这里我不想手动把3685条推特都挨个标记上主题（topic），因此无监督学习方法（PCA、LSI、LDA等）是比较可行的尝试方向。这里假设我们要抽取20个topic。

LSI的全名是Latent Semantic Indexing（隐性语义索引），它是基于SVD的方法，即用矩阵分解的方式将文本中的单词投射到向量空间中，从而抽取主题关键词（keyword）。我们通过gensim来实现LSI的分析。首先利用gensim中corpora模块建立BOW模型，再用models模块把它转化为tfidf模型（统计了单词的相对出现频率而不是出现次数），最后学习出一个LSI模型，从而得到20个topic的keyword，以及每个keyword在对应的余弦空间的投射。通过similarities模块，可以计算任意一个tweet文本与所有topic之间的余弦相似度。

LSI虽然可以提取topic，但是它只是一个纯粹的数学手段，并没有考虑单词间出现概率的影响；而LDA（Latent Dirichlet Allocation，隐含迪克特雷分析）则是基于三层贝叶斯概率模型的主题提取方法，包含了很多上下文之间的语义信息。但是，研究经验表示LDA对短文本的分析效果较差[5]，而推特可以说是短的不能再短的短文本了，怎么办？有NLP研究者开发了专门针对twitter的LDA方法（twitter-LDA）[5][6]，但代码是基于java的，无法直接应用；还有些文章的解决办法是将相同用户的推特合并在一起，再进行LDA分析[7]，可是这里所有的推特都是特朗普一个人发的，所以这个办法自然也不成立了。这里我的解决方案是根据LSI的计算结果，将每个tweet对所有topic的相似度（或者可以看成这个tweet在20维的topic空间的映射）作为feature，做一个Kmeans聚类，并将属于同一个聚类（因此也相对更加相似）的tweet拼接起来。不过这里还有一个问题，就是这样的聚类的效果是不均匀的，有的聚类的tweet数量很大，而很多聚类则只有1个tweet，这样简单合并的话就成了少数几个大段落和很多原封未动的单个tweet。因此，我只保留了聚类的中心和所有tweet对这些聚类中心的距离，然后对这些距离重新排序，并按照距离从近到远的顺序将所有tweet均匀的分配到每个聚类中心中去（算法复杂度为O(m*n log(n)，n为tweet个数，m为cluster个数）。设定的聚类个数为1000，也就是对于3685条tweet，每个聚类可以分配到3或4个tweet。

将tweet聚类并合并后，可以用scikit-learn里面的feature_extraction模块将新的文本集转化为tfidf模型，然后用decomposition模块里的LDA进行分析。利用工具包pyLDAvis可以打开一个网页，里面是LDA结果的可视化展示，截图如下:



网页的左侧用不同圈代表topic，圈的大小代表该topic的比重；横坐标和纵坐标分别为前两大的主成分（PC）。网页的右侧显示了所有词语的频率,与初步统计章节中的结果是基本吻合的（如果用鼠标点击某一个圈的话，还会显示该topic下同样词语的频率占比）。当然，尽管进行了聚类&合并处理，LDA的结果还是有一些杂乱之处；将LDA的结果整理成16个topic，每个topic保留8个keyword，列举如下：

Topic 1 (HRC): 'hillary', 'clinton', 'crooked', 'bernie', 'obama', 'years', 'campaign', 'bad'
Topic 2 (fakeNews): 'fake', 'news', 'dishonest', 'media', 'failing', 'big', 'story', 'said'
Topic 3 (MAGA): 'make', 'america', 'great', 'again', 'together', 'we', 'american', 'people'

Topic 4 (whitehouse): 'white', 'house', 'great', 'day', 'honor', 'today', 'meeting', 'senator'
Topic 5 (healthcare): 'obamacare', 'repeal', 'replace', 'disaster', 'crazy', 'bill', 'failed', 'reform'
Topic 6 (join): 'join', 'live', 'rally', 'tomorrow', 'tonight', 'tickets', 'today', 'speech'
Topic 7 (Korea&china): 'korea', 'north', 'south', 'china', 'trade', 'deficit', 'problem', 'president'
Topic 8 (police): 'law', 'enforcement', 'officers', 'police', 'executive', 'order', 'killed', 'victims'
Topic 9 (vote): 'trump', 'vote', 'poll', 'team', 'voters', 'final', 'americans', 'debate'
Topic 10 (jobs): 'jobs', 'bring', 'back', 'dollars', 'optimism', 'economic', 'market', 'companies'
Topic 11 (interview): 'enjoy', 'interviewed', 'tonight', 'looking', 'forward', 'interview', 'prime', 'minister'
Topic 12 (veterans): 'heroes', 'veterans', 'honor', 'today', 'act', 'announced', 'american', 'lives'
Topic 13 (terror): 'islamic', 'terror', 'radical', 'ban', 'tough', 'allowed', 'border', 'immigration'
Topic 14 (fbi): 'fbi', 'russia', 'cia', 'emails', 'director', 'illegally', 'investigation', 'server'
Topic 15 (mexico): 'mexico', 'wall', 'pay', 'trade', 'deficit', 'plant', 'crime', 'deal'
Topic 16 (refugee): 'syria', 'isis', 'syrian', 'refugees', 'immigrants', 'putin', 'rebels', 'ceasefire'

括号里的内容是我给Topic起的名字，可见这一年来比较热门的话题基本都涵盖了。上面pyLDAvis的网页截图上也标注了部分圆圈对应的话题名。

从主题分析的结果可以看出一些有趣的事情。例如，HRC（希拉里的姓名首字母简写）话题的keyword表明特朗普很喜欢用crooked这个词来形容希拉里，并且基本是如影随形的程度，而Bernie Sanders和Obama作为队友也经常在相似的语境里被喷；在fakeNews 话题里，dishonest media这个形容经常与fake

news交替出现，并且特朗普经常喜欢说他们的“big story”被证明“failing”了；谈论到Obamacare的时候，disaster、crazy、failed之类的形容词是跑不了了，而采取的行动就该是repeal、replace、reformed等。在网页左侧的分布图中，被开启嘴炮模式的HRC、fakenews、healthcare、terror等话题集中在第一象限里，而MAGA、whitehouse、vote、interview、join等表示较正面情绪的则在二四象限，至于jobs、korea&china、veterans、polices等感情较为复杂的一些话题则集体缩在靠近原点的区域中。

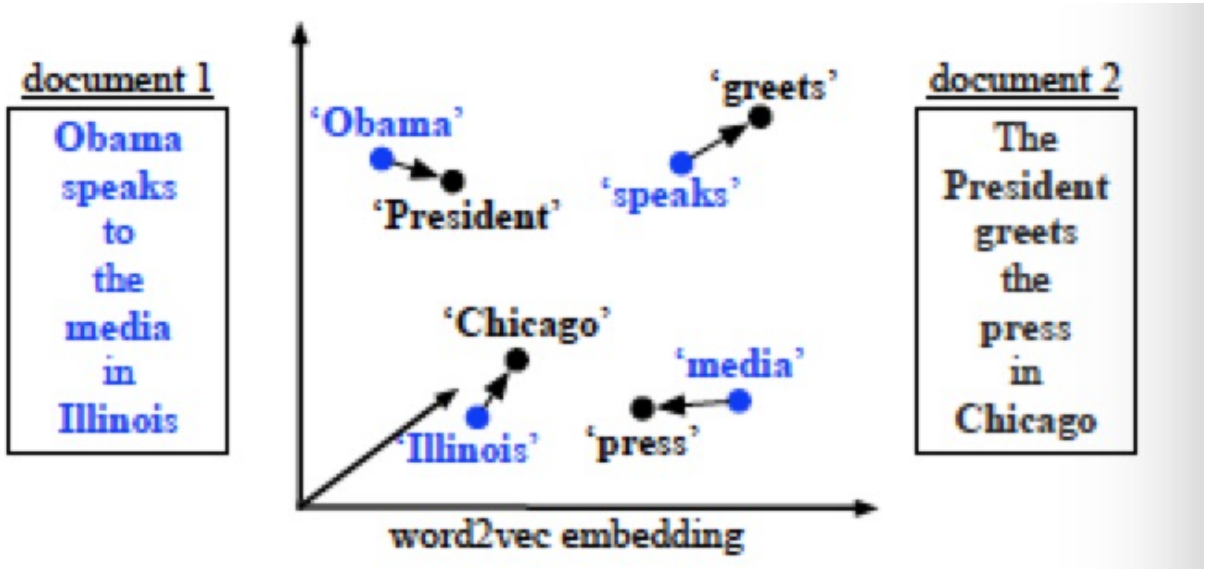
Word embedding和主题相似度

我们已经分析得到了16个主题，可是如何对tweet和主题之间进行更深层次的关联呢？我们需要对每个单词进行一些数字化的表达，而著名的word2vec就是实现word embedding的方法之一[8][9]。经过一定的学习过程，word2vec可以将不同的单词转化为高维空间中的向量，从而不同单词之间的相似程度可以用各自向量的距离来代表；得到的embedding向量还具备analogy性质，例如：“北京-中国 = 巴黎-法国”。这些向量可以用来进行后续的算法研究比如CNN、RNN之类。使用gensim.models中的Word2Vec模块[10]，用下载到的twitter数据作为输入，设定每个单词向量维度为500，很快就可以训练出一个小型模型；每个数据中出现过的单词都可以被返回一个相应的500维的向量。gensim还提供了其他一些好玩的功能，例如给定一个单词，返回与这个单词相似的前几个单词；这里用‘hillary’作为例子，模型表示最相似的单词依次是‘crooked’——相似度0.67；‘clinton’——相似度0.61；deleted——相似度0.55（这应该说的是她删除邮件的事情）；lying——相似度0.55，等等。确实，这个结果很特朗普。

虽然我们已经得到了自己的word2vec模型，但是不适合用它做后面的分析，因为训练的数据量太小了——仅仅三千多条twitter的文字量不太可能让机器学会人类遣词造句的规律。不过因为很多人已经用海量的数据训练过自己的word2vec模型，并且放在网上供人下载，因此我们这里可以用迁移学习的思想，直接使用这些预训练的模型来分析。这里我下载了一个用Google News训练的、包含300M个单词、维度为300的一个预训练模型[11]，并用gensim.models.keyedvectors模块将它读取到内存中。这样返回的单词向量就比较符合日常生活中的表述方式了（但是，用keyedvectors模块读取的模型不允许被继续训练——这是很大的一个遗憾，否则我们可以在预训练模型的基础

上，用特朗普的twitter数据进行重点训练，这样训练出来的向量就可以兼顾日常用法和特朗普的用词习惯了）。

现在我们有了一系列的以单词列表的形式存在的文本（初步分析后的tweet文本、topic keyword列表），也有了具体每个单词对应的空间向量值，怎么计算某条tweet与某个topic之间的相似度（即两个单词列表之间的相似度）呢？最近发表的文章[12]提供了一个有趣的思路。当比较两句话的相似度时，可以将每句的单词看成不同的节点，两个单词之间的向量距离看成两个节点之间的行走距离（word travel cost），则相似度的计算就可以想象成一个运筹学上的最小费用流问题：找到一系列从句1的某个节点流向句2的某个节点的“流”，并且它们的流量满足约束条件（1）从句1的任一节点流出的流量之和与句2的对应单词的词频相同；（2）向句2的任一节点流入的流量之和与句2的对应单词的词频相同；并且这些流导致的“travel cost”总和最小。显然，这是一个优化问题，优化得到的最小费用值即为两个文档之间的Word Mover's Distance（WMD）。WMD越小，两个文档的相似度越大。文章中给了一个形象的例子来展示计算WMD的过程：对于“Obama speaks to the media in Illinois”和“The President greets the press in Chicago”这两句话，首先过滤to、the、in等停止词，剩余的单词用于计算WMD如下图所示：



考虑到单词之间的相似度，显然优化得到的travel路径应该为Obama->President、speaks->greets、media->press和 Illinois,->Chicago，如图中箭头所示。

虽然WMD在理论上看起来很优美，但是每计算一次相似度就需要求解一次优化问题，这样会耗费大量的计算时间。因此，[12]中还提供了两个近似的计算方法，均能够得到WMD的下界。一种称为WCD（Word Centroid Distance），其实就是直接计算某句话中所有出现的单词的embedding向量的加权平均，这也是很多人采用的简便方法；另一种叫做Relaxed WMD （RWMD），就是把约束条件（2）去掉（Relax），这样就可以用直接比较的方法计算流量，省去了求解优化问题的过程。这里我采用了RWMD作为tweet与topic相似度的表征，感觉也是比较合适的；因为topic并不是一句话，而是一列关键词，因此我们比较关心的是tweet上的单词是否能够关联到某个keyword，而不在乎某个keyword是否没有或重复的被tweet上的单词关联到，这就规避了由于Relax第二个约束条件带来的精度损失问题。WMD和RWMD还有一个弱点是无法捕捉到句子中单词的排列次序；而topic的keyword list因为只是list，所以自然也不需要再担心次序问题。

当然，实际计算RWMD的过程也不像文章中的案例那么顺利，因为twitter作为短文本的社交媒体，里面有很多口语化的单词或有政治意义的专有名词，而这些token可能是不会存在于word2vec的模型中的，也就得不到对应的embedding向量。对于前者，一般不重要，计算RWMD的时候直接忽略掉就行了；而后者可能是决定一个tweet主题的关键部分，因此我建立了一个字典（python中是字典，其他语言里可能是个map），把这些专有名词映射到意思相近、存在于模型中的名词，计算的时候直接采用映射名词的向量就好了。例如，对于'obamacare'、'ocare'、'hcare'，都可以采用'healthcare'的embedding结果。

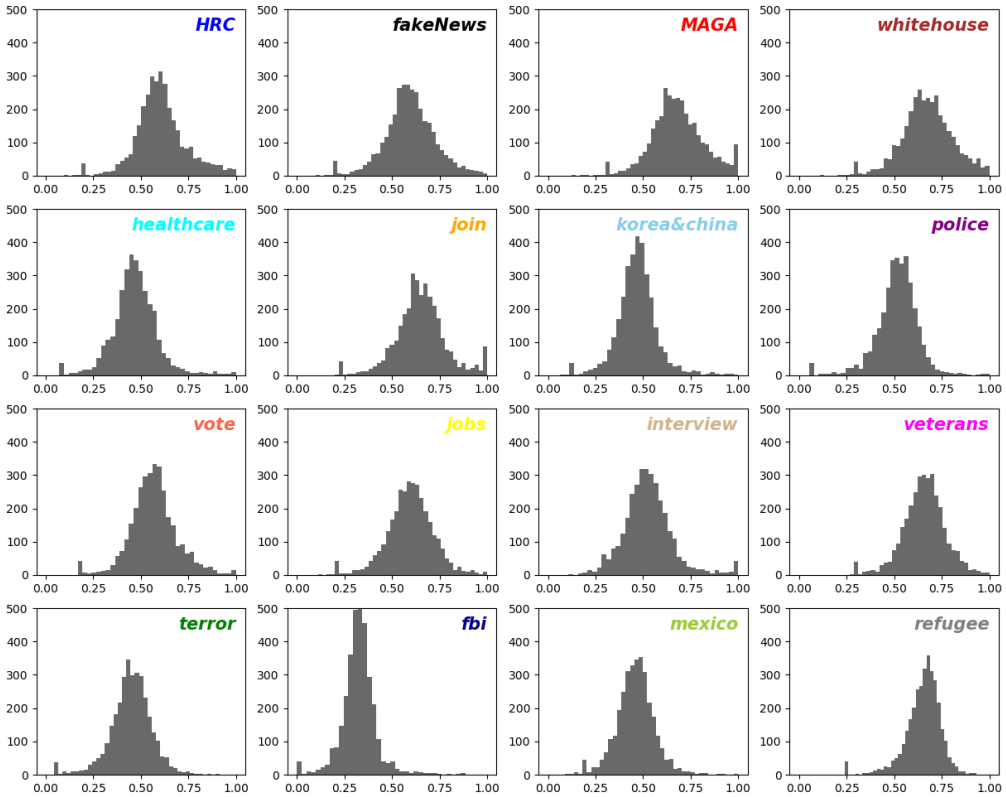
计算了RWMD之后，现在我们需要一个转换函数将RWMD转化为相似度（用百分比表示）；转换函数的具体形式为tr(RWMD) := 1-2/(1+exp(k(MD/RWMD-1)))，其中MD为所有RWMD数据中的最大值。显然，tr(0)->100%, tr(MD)=0%；k是人为设定的参数，代表同样大小的MD/RWMD被转化为相似度以后所接近0%的程度，暂时k设定为3.0。以特朗普在的一条tweet为例，原文为“The Fake News refuses to report the success of the first 6 months: S.C., surging economy & jobs,border & military security,ISIS & MS-13 etc.”，经过计算得到该tweet跟各topic之间的RWMD和相似度，见下表

	RWMD	相
HRC	3.10	59
fakeNews	2.83	71
MAGA	3.08	60
whitehouse	3.01	63
healthcare	3.46	43
join	3.04	63
korea&china	3.44	44
police	3.26	52
vote	3.28	52
jobs	2.91	68
interview	3.39	47
veterans	3.02	63
terror	3.29	51
fbi	3.65	35
mexico	3.46	43
refugee	2.98	65

这条tweet的内容是特朗普怒喷MSM们拒绝承认他执政前6个月的一系列成就：S.C.（意思存疑，可能说的是在South Carolina举办的共和党party？不过word2vec模型里是肯定没有关于'S.C.'这个词的向量的，这种情况我会跳过这个词）、经济复苏+工作增多、边境控制+反ISIS等，并按照惯例将MSM称为fake news。这里我们挑RWMD小于3.0的topic来看的话，与它比较相近的话题是fakeNews、jobs、refugee，还是比较符合主观理解的。

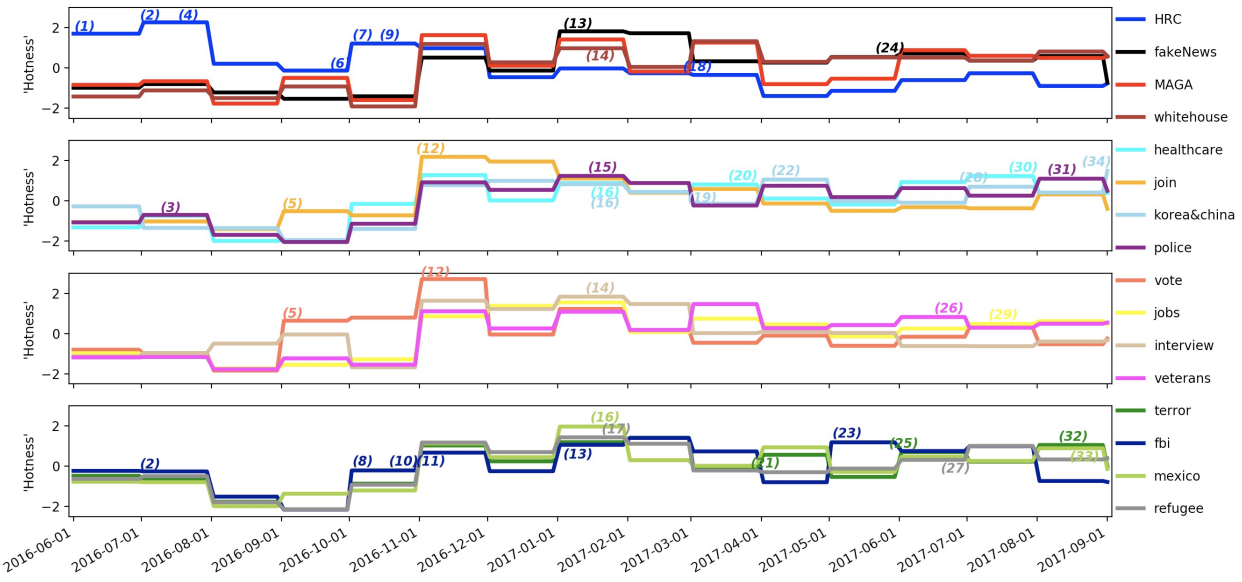
主题结果分析

首先对全部16个topic的相似度数据做一个histogram，如下图。可见不同topic的相似度分布的区别还是比较大的，例如fbi这个topic的相似度就普遍低一些，这倒不一定是跟这个topic有关的tweet数量少，而是它的keyword列表可能对应了不同事件（例如，可能谈论是fbi调查通俄门的事情（russia），也可能是希拉里的邮件门（emails、server），或者是fbi总监Comey的话题（director））。此外，很多topic都存在相似度等于100%的一个集中分布，说明一部分tweet是可以完美match这个tooic的。一个典型例子就是MAGA话题，特朗普的很多twitter就直接喊“make america great again”这个口号。



有了相似度的数据，接下来就可以以特朗普的twitter数据为依据，检查一下topic热度随着时间的变化趋势。由于相似度的分布区别很大，因此在绘图之前需要对它们先做一下standardize，所得到就是热度（Hotness）。为了绘图清晰，将16个topic分成4组进行展示如下；横坐标为时间，纵坐标为Hotness；图中的标记是一些比较match的标志性事件的发生情况。可见，相似度的结果还

是能够反应这一年多的时间里美国政治形势的变化的（当然，这首先是因为特朗普的twitter勤劳而忠实的评论了绝大部分的政治事件...）。



图中各标记对应的大事件：

(1) 2016.6月，特朗普、希拉里相继赢得初选。

(2) 2016.7.5，FBI重启对希拉里的邮件门调查，但随后宣布未发现犯罪事实。

(3) 2016.7月, Dallas、baton rouge、San Diego 分别发生警察被射杀事件。

(4) 2016.7.22，维基解密放出DNC泄漏邮件。

(5) 2016.9.6，总统竞选逐渐进入高潮，特朗普开始在各州进行巡回拉力（rally）。

(6) 2016.9.27，第一次总统大选辩论。

(7) 2016.10.7，特朗普“抓猫门”被爆出，随后特朗普将克林顿性侵案的受害者带到第二次总统辩论现场。

(8) 2016.10.7，维基解密爆出希拉里竞选经理Podesta的被盗账户里的邮件。

(9) 2016.10.9/19，第二次、第三次总统大选辩论。

(10) 2016.10.29，FBI在希拉里助理Huma Abedin的前夫电脑里发现了新的疑点，重启邮件门的调查。

(11) 2016.11.6，FBI总监Comey表示仍然没有发现希拉里新的罪证。同日维基解密放出DNC邮件的第二部分。

(12) 2016.11.8，大选日，特朗普以306票对232票战胜希拉里，获选总统。

(13) 2017.1月，FBI、CIA、NSA同时宣称俄罗斯干预了美国大选。特朗普称这些指控为“fake news”。
(14) 2017.1.20，特朗普宣布就职总统。
(15) 2017.1.20，特朗普就职的同时，美国几个城市爆发大规模游行示威，并发生多起人身冲突和伤害事件。
(16) 2017.1.20，就职后特朗普签署多份行政命令，尝试撤除Obamacare、退出TPP、重建墨西哥边境墙等。
(17) 2017.1.27，特朗普再次签署行政命令，对几个中东国家的难民颁布旅行禁令，后被联邦法院否决。
(18) 2017.3.4，特朗普指控奥巴马在去年竞选期间对其电话进行监控。
(19) 2017.3.6，朝鲜向日本方向发射4枚导弹。
(20) 2017.3.24，众议长Paul Ryan由于缺乏国会支持，决定推迟对Obamacare的表决，
(21) 2017.4.3，俄罗斯圣彼得堡恐袭。
(22) 2017.4.12，特朗普宣布不再将中国列为汇率操纵国；卡尔文森号和里根号先后驶向远东，应对朝鲜局势。
(23) 2017.5.9，FBI总监Comey被解职。
(24) 2017.5.28，特朗普称近日白宫流传的许多消息都是假新闻，并且质疑新闻来源。
(25) 2017.6.3，一辆卡车在伦敦大桥撞击行人。
(26) 2017.6.23，特朗普政府通过与老兵（veteran）有关的数个法案。
(27) 2017.6.26，法院部分批准了旅行禁令的执行；特朗普会见犯罪受害者家属，要求国会通过新的边境条例。
(28) 2017.7.4，朝鲜发射远程导弹，经过评估，射程可达阿拉斯加。
(29) 2017.7.17，特朗普发起“美国制造周”运动。
(30) 2017.7.31，众议院投票废除Obamacare。
(31) 2017.8.12，Charlottesville市的右翼游行造成多人伤亡。
(32) 2017.8.17，巴塞罗那恐袭。
(33) 2017.8.22，特朗普威胁如果国会不提供建墙费用，就将关闭国会
(34) 2017.9.3，朝鲜声称完成氢弹试验。

此外，我们还可以用hotness数据与原始数据中的其他维度相关联，尝试寻找新的结论。例如，原始数据中有每个tweet点赞数（ ‘favorite_count’ ）和转推数（ 'retweet_count'）的数据，我们姑且可以把它们当成是受欢迎程度的表

征（确切的说，这是在特朗普支持者中的受欢迎程度，而对于反对者而言，因为推特没有“踩”的数据，所以这些人的意见并没有包括在数据中）。通过设置一个hotness的阈值，当tweet与某个topic的相似度大于这个阈值时，我们就认为这个tweet“属于”该topic。这样，我们就可以统计每个topic平均的点赞数和转发数，以及转发/点赞的比例，如下表：

	平均点赞数	平均转发数	转
HRC	43818	10732	
fakeNews	56288	14424	
MAGA	40420	10064	
whitehouse	81288	18023	
healthcare	77913	13846	
join	70703	16296	
korea&china	46839	13097	
police	104106	16077	
vote	80555	14227	
jobs	101367	23200	
interview	126254	35167	
veterans	124870	24998	
terror	43796	11276	
fbi	63542	15002	
mexico	48399	16289	
refugee	126503	28850	

从表中可以看出，比较受欢迎的是refugee、interview、veterans和jobs等，看来难民管控和工作机会还是大家比较关心的话题；而对于攻击希拉里、朝鲜、中国、墨西哥这些比较富有争议的话题，则应者寥寥。转发数的趋势大体上与点

赞数是相似的，不过从转发/点赞比还是能看出一些小区别。我们都知道点赞是匿名的，而转发就会被推特好友看见，因此这可能体现了粉丝中的不同行为模式：例如对于vote、healthcare这些话题，转发数相比于点赞数就少了很多；而对于墨西哥、朝鲜、中国的话题，虽然点赞数很少，但转发的比例却非常高，大概能支持这些话题的都是最坚决的死忠，并不怕别人知道自己的政治倾向吧。

当然，我们还可以尝试用把topic看成feature，hotness数据作为输入，点赞数或转发数作为输出，这样就成了一个典型的监督学习的项目分别用sklearn里的Naive Bayes、SVM、adaboost、random forest等算法，做个3折交叉验证，大概能达到60-70%的精度。但是我们的数据量毕竟太小，而且实际上话题内容是显著随着时间变化的，因此学到的东西很可能大部分都是过拟合或噪点，这里就不展示结果了。

后记

除了监督学习，我们还可以把embedding的单词向量作为输入，跑一些深度学习模型例如RNN、CNN。label方面，除了点赞数和转发数，可以用来分析的维度还有特朗普发推的时间、发推用的手机/电脑等；当然，从这些角度分析的研究也已经很多了，例如[13]就是一个例子，还发表在了期刊上。。。此外本文中的主题分析用的都是预处理部分中被辨识为normal类型的token，而其他类型的token其实也可以用于分析中，例如hashtag、转发来源等。

此外，未来如果用一些爬虫工具爬到特朗普的全部twitter数据的话，就可以从空白开始训练word2vec模型，这样embedding向量就更符合特朗普的用语习惯。还可以分析他在竞选前喝竞选后的用语是否有变化，一定很有意思。

最后感谢张同学帮忙测试了部分监督学习方法，还有石同学提供了多条宝贵建议。

Python 代码 GitHub 地址：https://github.com/AaronJi/TrumpTwitterAnalysis
Package 要求：sklearn, gensim, nltk, tweepy, pyLDAvis, WordCloud, numpy, pandas

参考资料

[1] https://dev.twitter.com/overview/api/tweets
[2] http://www.tweepy.org
[3] https://github.com/bear/python-twitter
[4] Python 3 Text Processing with NLTK 3 Cookbook. Jacob Perkins, 2014
*[5] Comparing Twitter and T*现在自然语言处理（NLP）、机器学习（ML）等领域可谓是火爆异常，而随着社交网络的爆发式发展，自然也有很多研究者对社交媒体上的文本分析产生了兴趣。而说到社交媒体这块，玩的最溜、影响最大的可谓是现任美国总统特朗普（Trump）的推特（twitter）了，这哥们不仅常常一天狂发数条推特，还经常在上面直接宣布一些重大决定，或者耿直直接对政治对手进行攻击，人称“推特治国”。因此。用一些机器学习方面的算法来分析他的推特会不会产生一些有趣的结果呢？考虑到此， 这个练习项目就这么开始了。

数据获取

项目的第一步当然是下载足够的特朗普tweet数据。twitter提供了官方的API供用户用python语言与客户端交互；研究者只要提供自己的相应密钥信息，就可以通过它直接下载twitter相关的数据，具体方法可以参见相关介绍文档[1]。但是，官方API有一个很大的限制，就是下载的时候只能回溯同一个用户之前的3200条左右的tweet，再之前的就不给了。。。这让人有点失望，说好的大数据就这么没了。由于官方API的使用不是很方便，因此实际上我调用的是更上层的tweepy[2]，再通过它收集@readDonaldTrump这个账号的相关数据，包括发表过的推特的文本、发布时间、发布语言、点赞数、转发数、发布平台等。

下载程序参考[3]的架构，包括两部分功能：第一个部分负责下载并保存离当时最近的3000条tweet数据；第二个部分可以在稍晚时候运行，负责下载特朗普在之后新发的tweet。自从参加竞选以来，特朗普已经化身为发推狂，每天都要连发数条，所以在这个项目期间又极大的丰富了数据数量。最后用于分析的tweet从2016年6月2日5点开始，到2017年9月14日9点结束，共3685条tweet，保存为dataframe格式，如下图：

id	text	favorite_count	retweet_count	lang	source	created_y	created_m	created_d	created_h	created_min
908161523180285000	The "deplorables" came back to haunt Hillary.They expressed their feelings loud and clear. She spent bi	43818	10732	en	Twitter for iPhone	2017	9	14	2	52
908160218995068000	Crooked Hillary Clinton blames everybody (and every thing) but herself for her election loss. She lost th	56288	14424	en	Twitter for iPhone	2017	9	14	2	47
908154067658174000	China has a business tax rate of 15%. We should do everything possible to match them in order to win v	40420	10064	en	Twitter for iPhone	2017	9	14	2	22
907946177022369000	With Irma and Harvey devastation, Tax Cuts and Tax Reform is needed more than ever before. Go Cong	81288	18023	en	Twitter for iPhone	2017	9	13	12	36
907930425657626000	I will be traveling to Florida tomorrow to meet with our great Coast Guard, FEMA and many of the brav	77913	13846	en	Twitter for iPhone	2017	9	13	11	34
907928888587808000	The approval process for the biggest Tax Cut & Tax Reform package in the history of our country w	70703	16296	en	Twitter for iPhone	2017	9	13	11	28
907698529606541000	It was a great honor to welcome Prime Minister Najib Abdul Razak of Malaysia and his distinguished del	46839	13097	en	Twitter for iPhone	2017	9	12	20	12
907675638055743000	Congratulations to Eric & Lara on the birth of their son, Eric "Luke" Trump this morning! https://t.c	104106	16077	en	Twitter for iPhone	2017	9	12	18	41
907592460070768000	The devastation left by Hurricane Irma was far greater, at least in certain locations,than anyone though	80555	14227	en	Twitter for iPhone	2017	9	12	13	11

数据清洗

接下来我们需要对“text”这一列进行处理，也就是每条tweet的文本部分，其编码为utf-8。对于一般意义上的NLP分析，有很多语义分析工具可以使用；但是，twitter文本具有一些特殊的语法，因此这里还是需要做一些预处理工作。

首先假设文本中的每个单词都是用空格隔开的（事实上这也是英文的一般情况，不像中文还需要做分词工作）；这样可以用自然语言处理包NLTK[4]中的whitespaceTokenizer把tweet中每个用空格隔开的文本单元分成初步的token，再将这些token的字符全部转为小写。之后按照下列次序辨别一些特殊token：

7. 如果开头为“https”，这个token辨识为网站链接（link）
8. 如果开头为“RT”，则该token为转发的tweet（retweet）的固定开头
9. 如果开头为“@”，则为推特的用户名
10. 如果开头为“#”，则代表“hashtag”（twitter中用来标注线索主题的标签）
11. 如果开头为“&”，则基本上是tweet中的“&”符号，在utf-8的文本中以“&”代表（目前暂未发现反例）
12. 通过一个正则表达式，辨识token是否为email地址

排除掉这些特殊token之后，对于剩余token再用Tokenizer再做一次分词（我用的WordPunctTokenizer），这样可以拆分那些没有被空格隔开的、被标点符号、连字符连接的多词token。最后对每个token再做一次检查，如果是全部由拉丁字母和数字组成的，那基本可以认为是普通类型的token（标记成normal）；否则则说明它包含一些特殊字符，例如原本tweet里的表情符号被强制转换成utf-8文本之后，会变成一些乱码；或者是一些外国语言的人名之类

的，这些token统一标记成“special”类型。在后续的NLP分析中，只考虑normal类型的token。

这样，每个tweet就转化为了一个token组成的list，每个token有自己对应的类型。同时，我们还对tweet的整体做了一些识别；比如，tweet的首个token是“RT”的话，则第二个token自动对应转发来源，且整个tweet都标记成“retweet”，这样后续的一些分析可能就不再考虑这种转发的推特（例如，retweet的点赞数一般会明显小于原创的tweet）；可以把在单/双引号内部的token全部标记成“被引用状态”，后面可以决定是否考虑这些内容（因为引用的可能是别人的话；但其实也很难说，因为也可能只是表示讽刺之类的）；还可以根据token辨识的结果生成一个不含特殊字符的“干净版”的文本，然后用NLTK的sent_tokenize函数进行分句，从而可以进行每个sentence的分析等等。

这里我们随机挑选一个tweet，检验一下我们的文本分析的结果，例如特朗普在今年6月22日发的一条tweet：

注意下方插入的视频，在转化为utf-8文本数据的时候，它会变成一个链接。我们的分析结果为（括号中为辨识到的token类型）：

we(normal) '(special) re(normal) all(normal) thinking(normal) of(normal) you(normal) stevescalise(tweetUser) teamscalise(Hashtag) https://t.co/yqf6exhm7x(link)

初步统计

数据清洗完成之后，我们可以进行一些基本的分析了，比如建立一个简单的词袋（bag of words，BOW）模型。这里我们只统计原创的（非转发）tweet

中类型为normal的tokens，并且过滤掉停止词（stopwords，如没有实际语义的to、for、a等）、单纯的数字和部分（无意义）的地名等。统计得到数据中共有57589个单词、31503个非停止词、5206个独特的单词。使用频率最高的几个单词为great（516次）、thank（397次）、hillary（362次）、clinton（261次）、people（241次）、trump（209次）、america（198次）等，很符合我们对trump用语的一贯印象。最后将词袋里的单词画成词云（word cloud），作为一个形象化的展示：

无监督学习

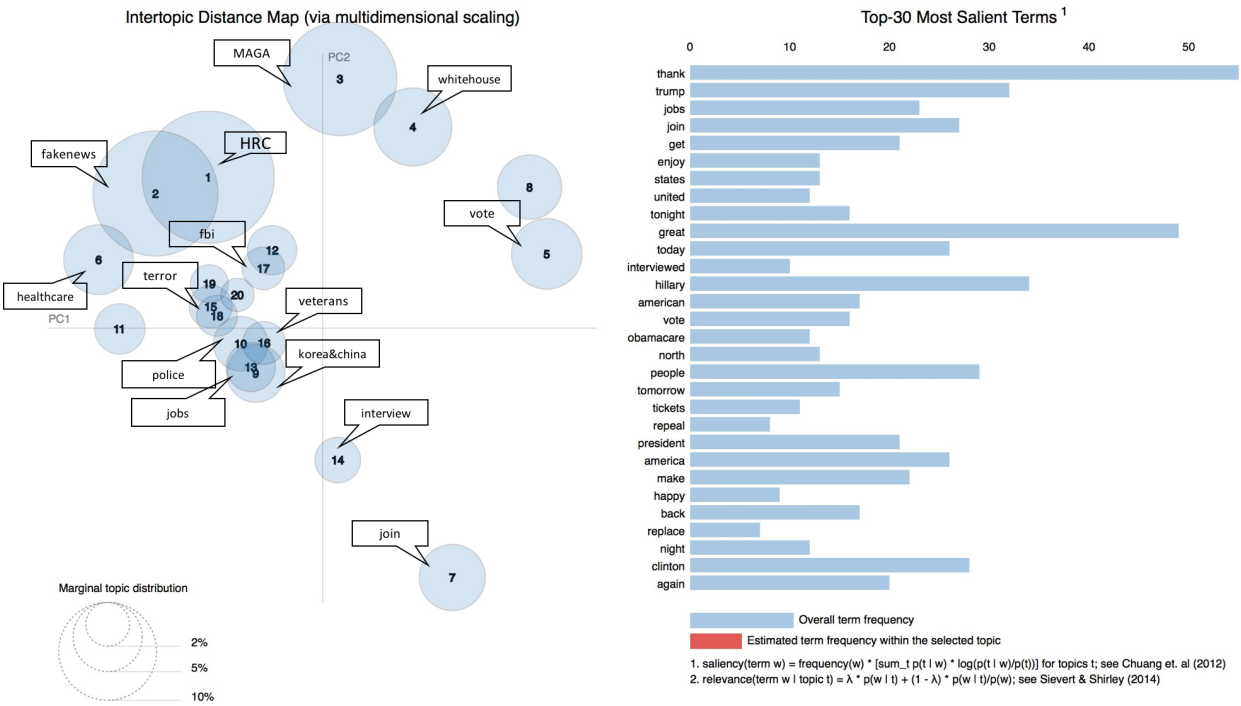
主题分析是NLP的一个重要组成部分，在这里也是一个很有意思的研究方向：特朗普的每条tweet可能分属不同的主题，比如竞选拉票、喷政治对手、喷外国、讨论工作问题、讨论医保问题等等。因为这里我不想手动把3685条推特都挨个标记上主题（topic），因此无监督学习方法（PCA、LSI、LDA等）是比较可行的尝试方向。这里假设我们要抽取20个topic。

LSI的全名是Latent Semantic Indexing（隐性语义索引），它是基于SVD的方法，即用矩阵分解的方式将文本中的单词投射到向量空间中，从而抽取主题关键词（keyword）。我们通过gensim来实现LSI的分析。首先利用gensim中corpora模块建立BOW模型，再用models模块把它转化为tfidf模型（统计了单词的相对出现频率而不是出现次数），最后学习出一个LSI模型，从而得到20个

topic的keyword，以及每个keyword在对应的余弦空间的投射。通过similarities模块，可以计算任意一个tweet文本与所有topic之间的余弦相似度。

LSI虽然可以提取topic，但是它只是一个纯粹的数学手段，并没有考虑单词间出现概率的影响；而LDA（Latent Dirichlet Allocation，隐含迪克特雷分析）则是基于三层贝叶斯概率模型的主题提取方法，包含了很多上下文之间的语义信息。但是，研究经验表示LDA对短文本的分析效果较差[5]，而推特可以说是短的不能再短的短文本了，怎么办？有NLP研究者开发了专门针对twitter的LDA方法（twitter-LDA）[5][6]，但代码是基于java的，无法直接应用；还有些文章的解决办法是将相同用户的推特合并在一起，再进行LDA分析[7]，可是这里所有的推特都是特朗普一个人发的，所以这个办法自然也不成立了。这里我的解决方案是根据LSI的计算结果，将每个tweet对所有topic的相似度（或者可以看成这个tweet在20维的topic空间的映射）作为feature，做一个Kmeans聚类，并将属于同一个聚类（因此也相对更加相似）的tweet拼接起来。不过这里还有一个问题，就是这样的聚类的效果是不均匀的，有的聚类的tweet数量很大，而很多聚类则只有1个tweet，这样简单合并的话就成了少数几个大段落和很多原封未动的单个tweet。因此，我只保留了聚类的中心和所有tweet对这些聚类中心的距离，然后对这些距离重新排序，并按照距离从近到远的顺序将所有tweet均匀的分配到每个聚类中心中去（算法复杂度为O(m*n log(n)，n为tweet个数，m为cluster个数）。设定的聚类个数为1000，也就是对于3685条tweet，每个聚类可以分配到3或4个tweet。

将tweet聚类并合并后，可以用scikit-learn里面的feature_extraction模块将新的文本集转化为tfidf模型，然后用decomposition模块里的LDA进行分析。利用工具包pyLDAvis可以打开一个网页，里面是LDA结果的可视化展示，截图如下:



网页的左侧用不同圈代表topic，圈的大小代表该topic的比重；横坐标和纵坐标分别为前两大的主成分（PC）。网页的右侧显示了所有词语的频率,与初步统计章节中的结果是基本吻合的（如果用鼠标点击某一个圈的话，还会显示该topic下同样词语的频率占比）。当然，尽管进行了聚类&合并处理，LDA的结果还是有一些杂乱之处；将LDA的结果整理成16个topic，每个topic保留8个keyword，列举如下：

Topic 1 (HRC): 'hillary', 'clinton', 'crooked', 'bernie', 'obama', 'years', 'campaign', 'bad'
Topic 2 (fakeNews): 'fake', 'news', 'dishonest', 'media', 'failing', 'big', 'story', 'said'
Topic 3 (MAGA): 'make', 'america', 'great', 'again', 'together', 'we', 'american', 'people'
Topic 4 (whitehouse): 'white', 'house', 'great', 'day', 'honor', 'today', 'meeting', 'senator'
Topic 5 (healthcare): 'obamacare', 'repeal', 'replace', 'disaster', 'crazy', 'bill', 'failed', 'reform'
Topic 6 (join): 'join', 'live', 'rally', 'tomorrow', 'tonight', 'tickets', 'today', 'speech'

Topic 7 (Korea&china): 'korea', 'north', 'south', 'china', 'trade', 'deficit', 'problem', 'president'
Topic 8 (police): 'law', 'enforcement', 'officers', 'police', 'executive', 'order', 'killed', 'victims'
Topic 9 (vote): 'trump', 'vote', 'poll', 'team', 'voters', 'final', 'americans', 'debate'
Topic 10 (jobs): 'jobs', 'bring', 'back', 'dollars', 'optimism', 'economic', 'market', 'companies'
Topic 11 (interview): 'enjoy', 'interviewed', 'tonight', 'looking', 'forward', 'interview', 'prime', 'minister'
Topic 12 (veterans): 'heroes', 'veterans', 'honor', 'today', 'act', 'announced', 'american', 'lives'
Topic 13 (terror): 'islamic', 'terror', 'radical', 'ban', 'tough', 'allowed', 'border', 'immigration'
Topic 14 (fbi): 'fbi', 'russia', 'cia', 'emails', 'director', 'illegally', 'investigation', 'server'
Topic 15 (mexico): 'mexico', 'wall', 'pay', 'trade', 'deficit', 'plant', 'crime', 'deal'
Topic 16 (refugee): 'syria', 'isis', 'syrian', 'refugees', 'immigrants', 'putin', 'rebels', 'ceasefire'

括号里的内容是我给Topic起的名字，可见这一年来比较热门的话题基本都涵盖了。上面pyLDAvis的网页截图上也标注了部分圆圈对应的话题名。

从主题分析的结果可以看出一些有趣的事情。例如，HRC（希拉里的姓名首字母简写）话题的keyword表明特朗普很喜欢用crooked这个词来形容希拉里，并且基本是如影随形的程度，而Bernie Sanders和Obama作为队友也经常在相似的语境里被喷；在fakeNews 话题里，dishonest media这个形容经常与fake news 交替出现，并且特朗普经常喜欢说他们的“big story”被证明“failing”了；谈论到Obamacare的时候，disaster、crazy、failed之类的形容词是跑不了了，而采取的行动就该是repeal、replace、reformed等。在网页左侧的分布图中，被开启嘴炮模式的HRC、fakenews、healthcare、terror等话题集中在第一象限里，而MAGA、whitehouse、vote、interview、join等

表示较正面情绪的则在二四象限，至于jobs、korea&china、veterans、polices等感情较为复杂的一些话题则集体缩在靠近原点的区域中。

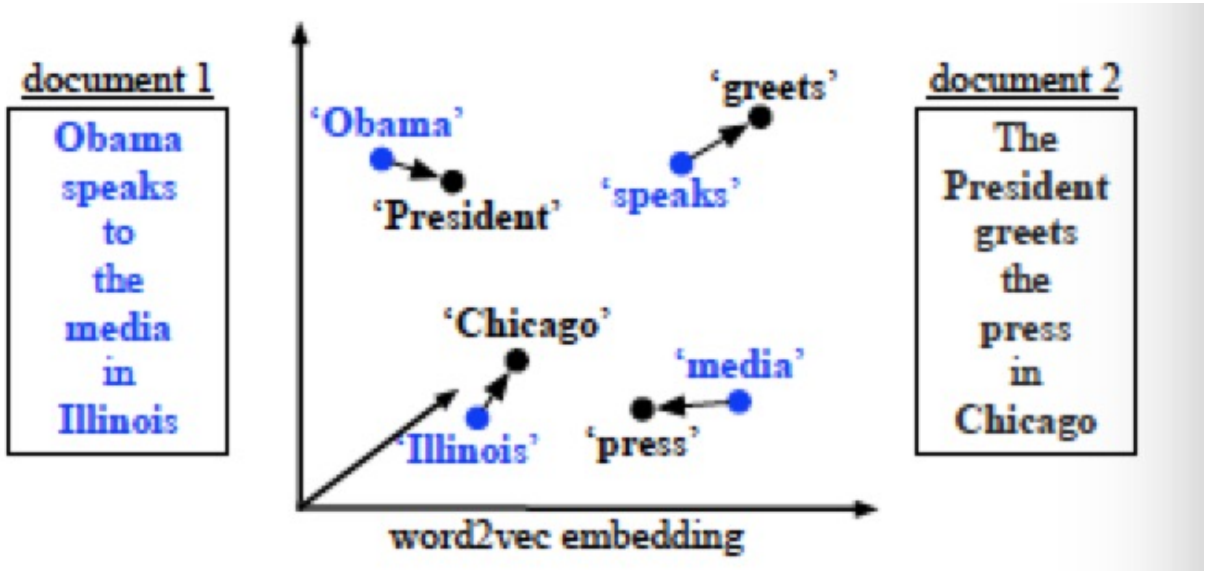
Word embedding和主题相似度

我们已经分析得到了16个主题，可是如何对tweet和主题之间进行更深层次的关联呢？ 我们需要对每个单词进行一些数字化的表达，而著名的word2vec就是实现word embedding的方法之一[8][9]。经过一定的学习过程，word2vec可以将不同的单词转化为高维空间中的向量，从而不同单词之间的相似程度可以用各自向量的距离来代表；得到的embedding向量还具备analogy性质，例如：“北京-中国 = 巴黎-法国”。这些向量可以用来进行后续的算法研究比如CNN、RNN之类。使用gensim.models中的Word2Vec模块[10]，用下载到的twitter数据作为输入，设定每个单词向量维度为500，很快就可以训练出一个小型模型；每个数据中出现过的单词都可以被返回一个相应的500维的向量。gensim还提供了其他一些好玩的功能，例如给定一个单词，返回与这个单词相似的前几个单词；这里用‘hillary’作为例子，模型表示最相似的单词依次是‘crooked’——相似度0.67；‘clinton’——相似度0.61；deleted——相似度0.55（这应该说的是她删除邮件的事情）；lying——相似度0.55，等等。确实，这个结果很特朗普。

虽然我们已经得到了自己的word2vec模型，但是不适合用它做后面的分析，因为训练的数据量太小了——仅仅三千多条twitter的文字量不太可能让机器学会人类遣词造句的规律。不过因为很多人已经用海量的数据训练过自己的word2vec模型，并且放在网上供人下载，因此我们这里可以用迁移学习的思想，直接使用这些预训练的模型来分析。这里我下载了一个用Google News训练的、包含300M个单词、维度为300的一个预训练模型[11]，并用gensim.models.keyedvectors模块将它读取到内存中。这样返回的单词向量就比较符合日常生活中的表述方式了（但是，用keyedvectors模块读取的模型不允许被继续训练——这是很大的一个遗憾，否则我们可以在预训练模型的基础上，用特朗普的twitter数据进行重点训练，这样训练出来的向量就可以兼顾日常用法和特朗普的用词习惯了）。

现在我们有了一系列的以单词列表的形式存在的文本（初步分析后的tweet文本、topic keyword列表），也有了具体每个单词对应的空间向量值，怎么计算某条tweet与某个topic之间的相似度（即两个单词列表之间的相似度）呢？最

近发表的文章[12]提供了一个有趣的思路。当比较两句话的相似度时，可以将每句的单词看成不同的节点，两个单词之间的向量距离看成两个节点之间的行走距离（word travel cost），则相似度的计算就可以想象成一个运筹学上的最小费用流问题：找到一系列从句1的某个节点流向句2的某个节点的“流”，并且它们的流量满足约束条件（1）从句1的任一节点流出的流量之和与句2的对应单词的词频相同；（2）向句2的任一节点流入的流量之和与句2的对应单词的词频相同；并且这些流导致的“travel cost”总和最小。显然，这是一个优化问题，优化得到的最小费用值即为两个文档之间的Word Mover's Distance（WMD）。WMD越小，两个文档的相似度越大。文章中给了一个形象的例子来展示计算WMD的过程：对于“Obama speaks to the media in Illinois”和“The President greets the press in Chicago”这两句话，首先过滤to、the、in等停止词，剩余的单词用于计算WMD如下图所示：



考虑到单词之间的相似度，显然优化得到的travel路径应该为Obama->President、speaks->greets、media->press和 Illinois,->Chicago，如图中箭头所示。

虽然WMD在理论上看起来很优美，但是每计算一次相似度就需要求解一次优化问题，这样会耗费大量的计算时间。因此，[12]中还提供了两个近似的计算方法，均能够得到WMD的下界。一种称为WCD（Word Centroid Distance），其实就是直接计算某句话中所有出现的单词的embedding向量的加权平均，这也是很多人采用的简便方法；另一种叫做Relaxed WMD （RWMD），就是把

约束条件（2）去掉（Relax），这样就可以用直接比较的方法计算流量，省去了求解优化问题的过程。这里我采用了RWMD作为tweet与topic相似度的表征，感觉也是比较合适的；因为topic并不是一句话，而是一列关键词，因此我们比较关心的是tweet上的单词是否能够关联到某个keyword，而不在乎某个keyword是否没有或重复的被tweet上的单词关联到，这就规避了由于Relax第二个约束条件带来的精度损失问题。WMD和RWMD还有一个弱点是无法捕捉到句子中单词的排列次序；而topic的keyword list因为只是list，所以自然也不需要再担心次序问题。

当然，实际计算RWMD的过程也不像文章中的案例那么顺利，因为twitter作为短文本的社交媒体，里面有很多口语化的单词或有政治意义的专有名词，而这些token可能是不会存在于word2vec的模型中的，也就得不到对应的embedding向量。对于前者，一般不重要，计算RWMD的时候直接忽略掉就行了；而后者可能是决定一个tweet主题的关键部分，因此我建立了一个字典（python中是字典，其他语言里可能是个map），把这些专有名词映射到意思相近、存在于模型中的名词，计算的时候直接采用映射名词的向量就好了。例如，对于'obamacare'、'ocare'、'hcare'，都可以采用'healthcare'的embedding结果。

计算了RWMD之后，现在我们需要一个转换函数将RWMD转化为相似度（用百分比表示）；转换函数的具体形式为tr(RWMD) := 1-2/(1+exp(k(MD/RWMD-1)))，其中MD为所有RWMD数据中的最大值。显然，tr(0)->100%, tr(MD)=0%；k是人为设定的参数，代表同样大小的MD/RWMD被转化为相似度以后所接近0%的程度，暂时k设定为3.0。以特朗普在的一条tweet为例，原文为“The Fake News refuses to report the success of the first 6 months: S.C., surging economy & jobs,border & military security,ISIS & MS-13 etc.”，经过计算得到该tweet跟各topic之间的RWMD和相似度，见下表

	RWMD	相
HRC	3.10	59
fakeNews	2.83	71
MAGA	3.08	60

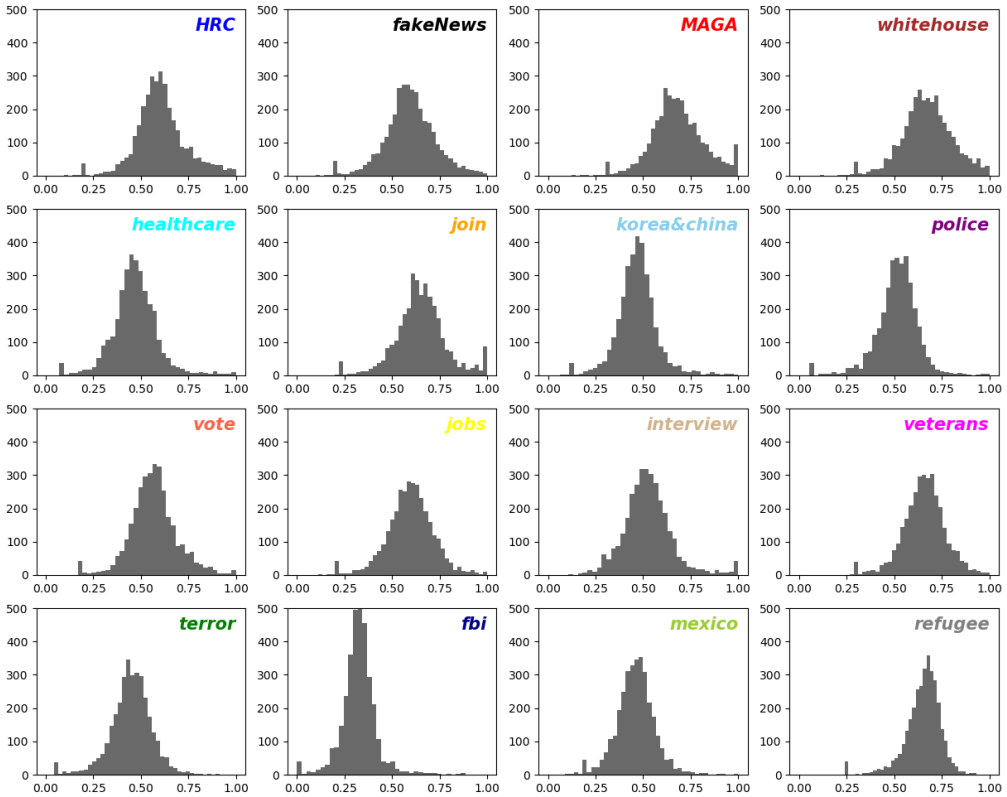
whitehouse	3.01	63
healthcare	3.46	43
join	3.04	63
korea&china	3.44	44
police	3.26	52
vote	3.28	52
jobs	2.91	68
interview	3.39	47
veterans	3.02	63
terror	3.29	51
fbi	3.65	35
mexico	3.46	43
refugee	2.98	65

这条tweet的内容是特朗普怒喷MSM们拒绝承认他执政前6个月的一系列成就：S.C.（意思存疑，可能说的是在South Carolina举办的共和党party？不过word2vec模型里是肯定没有关于'S.C.'这个词的向量的，这种情况我会跳过这个词）、经济复苏+工作增多、边境控制+反ISIS等，并按照惯例将MSM称为fake news。这里我们挑RWMD小于3.0的topic来看的话，与它比较相近的话题是fakeNews、jobs、refugee，还是比较符合主观理解的。

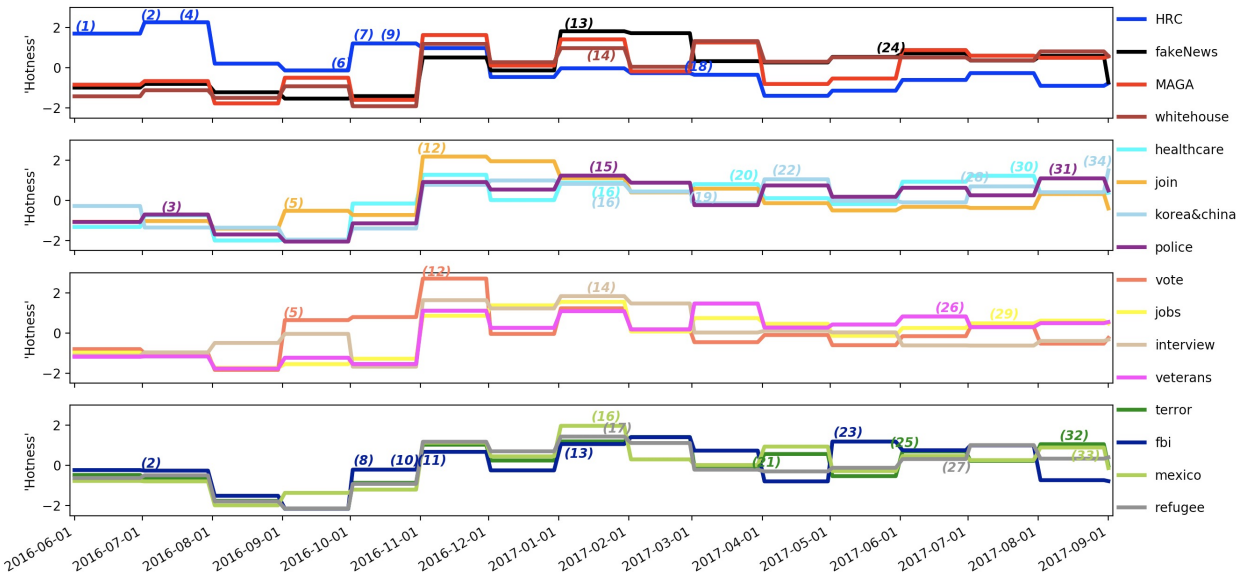
主题结果分析

首先对全部16个topic的相似度数据做一个histogram，如下图。可见不同topic的相似度分布的区别还是比较大的，例如fbi这个topic的相似度就普遍低一些，这倒不一定是跟这个topic有关的tweet数量少，而是它的keyword列表可能对应了不同事件（例如，可能谈论是fbi调查通俄门的事情（russia），也可能是希拉里的邮件门（emails、server），或者是fbi总监Comey的话题（director））。此外，很多topic都存在相似度等于100%的一个集中分布，说

明一部分tweet是可以完美match这个tooic的。一个典型例子就是MAGA话题，特朗普的很多twitter就直接喊“make america great again”这个口号。



有了相似度的数据，接下来就可以以特朗普的twitter数据为依据，检查一下topic热度随着时间的变化趋势。由于相似度的分布区别很大，因此在绘图之前需要对它们先做一下standardize，所得到就是热度（Hotness）。为了绘图清晰，将16个topic分成4组进行展示如下；横坐标为时间，纵坐标为Hotness；图中的标记是一些比较match的标志性事件的发生情况。可见，相似度的结果还是能够反应这一年多的时间里美国政治形势的变化的（当然，这首先是因为特朗普的twitter勤劳而忠实的评论了绝大部分的政治事件...）。



图中各标记对应的大事件：

(1) 2016.6月，特朗普、希拉里相继赢得初选。

(2) 2016.7.5，FBI重启对希拉里的邮件门调查，但随后宣布未发现犯罪事实。

(3) 2016.7月, Dallas、baton rouge、San Diego 分别发生警察被射杀事件。

(4) 2016.7.22，维基解密放出DNC泄漏邮件。

(5) 2016.9.6，总统竞选逐渐进入高潮，特朗普开始在各州进行巡回拉力（rally）。

(6) 2016.9.27，第一次总统大选辩论。

(7) 2016.10.7，特朗普“抓猫门”被爆出，随后特朗普将克林顿性侵案的受害者带到第二次总统辩论现场。

(8) 2016.10.7，维基解密爆出希拉里竞选经理Podesta的被盗账户里的邮件。

(9) 2016.10.9/19，第二次、第三次总统大选辩论。

(10) 2016.10.29，FBI在希拉里助理Huma Abedin的前夫电脑里发现了新的疑点，重启邮件门的调查。

(11) 2016.11.6，FBI总监Comey表示仍然没有发现希拉里新的罪证。同日维基解密放出DNC邮件的第二部分。

(12) 2016.11.8，大选日，特朗普以306票对232票战胜希拉里，获选总统。

(13) 2017.1月，FBI、CIA、NSA同时宣称俄罗斯干预了美国大选。特朗普称这些指控为“fake news”。

(14) 2017.1.20，特朗普宣布就职总统。

(15) 2017.1.20，特朗普就职的同时，美国几个城市爆发大规模游行示威，并发生多起人身冲突和伤害事件。
(16) 2017.1.20，就职后特朗普签署多份行政命令，尝试撤除Obamacare、退出TPP、重建墨西哥边境墙等。
(17) 2017.1.27，特朗普再次签署行政命令，对几个中东国家的难民颁布旅行禁令，后被联邦法院否决。
(18) 2017.3.4，特朗普指控奥巴马在去年竞选期间对其电话进行监控。
(19) 2017.3.6，朝鲜向日本方向发射4枚导弹。
(20) 2017.3.24，众议长Paul Ryan由于缺乏国会支持，决定推迟对Obamacare的表决，
(21) 2017.4.3，俄罗斯圣彼得堡恐袭。
(22) 2017.4.12，特朗普宣布不再将中国列为汇率操纵国；卡尔文森号和里根号先后驶向远东，应对朝鲜局势。
(23) 2017.5.9，FBI总监Comey被解职。
(24) 2017.5.28，特朗普称近日白宫流传的许多消息都是假新闻，并且质疑新闻来源。
(25) 2017.6.3，一辆卡车在伦敦大桥撞击行人。
(26) 2017.6.23，特朗普政府通过与老兵（veteran）有关的数个法案。
(27) 2017.6.26，法院部分批准了旅行禁令的执行；特朗普会见犯罪受害者家属，要求国会通过新的边境条例。
(28) 2017.7.4，朝鲜发射远程导弹，经过评估，射程可达阿拉斯加。
(29) 2017.7.17，特朗普发起“美国制造周”运动。
(30) 2017.7.31，众议院投票废除Obamacare。
(31) 2017.8.12，Charlottesville市的右翼游行造成多人伤亡。
(32) 2017.8.17，巴塞罗那恐袭。
(33) 2017.8.22，特朗普威胁如果国会不提供建墙费用，就将关闭国会
(34) 2017.9.3，朝鲜声称完成氢弹试验。

此外，我们还可以用hotness数据与原始数据中的其他维度相关联，尝试寻找新的结论。例如，原始数据中有每个tweet点赞数（ ‘favorite_count’ ）和转推数（'retweet_count'）的数据，我们姑且可以把它们当成是受欢迎程度的表征（确切的说，这是在特朗普支持者中的受欢迎程度，而对于反对者而言，因为推特没有“踩”的数据，所以这些人的意见并没有包括在数据中）。通过设置一个hotness的阈值，当tweet与某个topic的相似度大于这个阈值时，我们就认为

这个tweet“属于”该topic。这样，我们就可以统计每个topic平均的点赞数和转发数，以及转发/点赞的比例，如下表：

	平均点赞数	平均转发数	转
HRC	43818	10732	
fakeNews	56288	14424	
MAGA	40420	10064	
whitehouse	81288	18023	
healthcare	77913	13846	
join	70703	16296	
korea&china	46839	13097	
police	104106	16077	
vote	80555	14227	
jobs	101367	23200	
interview	126254	35167	
veterans	124870	24998	
terror	43796	11276	
fbi	63542	15002	
mexico	48399	16289	
refugee	126503	28850	

从表中可以看出，比较受欢迎的是refugee、interview、veterans和jobs等，看来难民管控和工作机会还是大家比较关心的话题；而对于攻击希拉里、朝鲜、中国、墨西哥这些比较富有争议的话题，则应者寥寥。转发数的趋势大体上与点赞数是相似的，不过从转发/点赞比还是能看出一些小区别。我们都知道点赞是匿名的，而转发就会被推特好友看见，因此这可能体现了粉丝中的不同行为模式：例如对于vote、healthcare这些话题，转发数相比于点赞数就少了很多；

而对于墨西哥、朝鲜、中国的话题，虽然点赞数很少，但转发的比例却非常高，大概能支持这些话题的都是最坚决的死忠，并不怕别人知道自己的政治倾向吧。

当然，我们还可以尝试用把topic看成feature，hotness数据作为输入，点赞数或转发数作为输出，这样就成了一个典型的监督学习的项目分别用sklearn里的Naive Bayes、SVM、adaboost、random forest等算法，做个3折交叉验证，大概能达到60-70%的精度。但是我们的数据量毕竟太小，而且实际上话题内容是显著随着时间变化的，因此学到的东西很可能大部分都是过拟合或噪点，这里就不展示结果了。

后记

除了监督学习，我们还可以把embedding的单词向量作为输入，跑一些深度学习模型例如RNN、CNN。label方面，除了点赞数和转发数，可以用来分析的维度还有特朗普发推的时间、发推用的手机/电脑等；当然，从这些角度分析的研究也已经很多了，例如[13]就是一个例子，还发表在了期刊上。。。此外本文中的主题分析用的都是预处理部分中被辨识为normal类型的token，而其他类型的token其实也可以用于分析中，例如hashtag、转发来源等。

此外，未来如果用一些爬虫工具爬到特朗普的全部twitter数据的话，就可以从空白开始训练word2vec模型，这样embedding向量就更符合特朗普的用语习惯。还可以分析他在竞选前喝竞选后的用语是否有变化，一定很有意思。

最后感谢张同学帮忙测试了部分监督学习方法，还有石同学提供了多条宝贵建议。

Python 代码 GitHub 地址：https://github.com/AaronJi/TrumpTwitterAnalysis
Package 要求：sklearn, gensim, nltk, tweepy, pyLDAvis, WordCloud, numpy, pandas

参考资料

[1] https://dev.twitter.com/overview/api/tweets

[2] http://www.tweepy.org

[3] https://github.com/bear/python-twitter

[4] Python 3 Text Processing with NLTK 3 Cookbook. Jacob Perkins, 2014
[5] Comparing Twitter and Traditional Media using Topic Models. Zhao, Jiang, Weng, He, Lim, Research Collection School of Information Systems, 2011
[6] https://www.slideshare.net/akshayubhat/twitter-lda
[7] Empirical Study of Topic Modeling in Twitter. Hong, Davison, 1st Workshop on Social Media Analytics, 2010
[8] Efficient Estimation of Word Representations in Vector Space. Mikolov, Chen, Corrado, Dean, arXiv, 2013
[9] Distributed Representations of Words and Phrases and their Compositionality. Serban, Sordoni, Bengio, Courville, Pineau, Advances in Neural Information Processing System 26, 2013
[10] https://radimrehurek.com/gensim/models/word2vec.html
[11] https://github.com/3Top/word2vec-api
[12] From Word Embeddings To Document Distances. Kusner, Sun, Kolkin, Weinberger, Proceedings of the 32nd International Conference on Machine Learning, 2015
[13] Twitter as a means to study temporal behaviour. Roenneberg, Current Biology Cb, 2017raditional Media using Topic Models. Zhao, Jiang, Weng, He, Lim, Research Collection School of Information Systems, 2011
[6] https://www.slideshare.net/akshayubhat/twitter-lda
[7] Empirical Study of Topic Modeling in Twitter. Hong, Davison, 1st Workshop on Social Media Analytics, 2010
[8] Efficient Estimation of Word Representations in Vector Space. Mikolov, Chen, Corrado, Dean, arXiv, 2013
[9] Distributed Representations of Words and Phrases and their Compositionality. Serban, Sordoni, Bengio, Courville, Pineau, Advances in Neural Information Processing System 26, 2013
[10] https://radimrehurek.com/gensim/models/word2vec.html
[11] https://github.com/3Top/word2vec-api

[12] From Word Embeddings To Document Distances. Kusner, Sun, Kolkin, Weinberger, Proceedings of the 32nd International Conference on Machine Learning, 2015

[13] Twitter as a means to study temporal behaviour. Roenneberg, Current Biology Cb, 2017