# 美国总统Twitter辩论

## 目录



## 1 引言

2020年的美国总统大选引来了同学们的不少关注，一方面是不同总统对华人的态度有所不同，可能会影响同学们求学的难度，另一方面是两位总统在Twitter上的风格截然不同，Donald Trump从2016年当选总统以来，一直被戏称为“Twitter治国”，他的tweet带有鲜明的个人情感因素以及特征，与总统一词的印象大相径庭，最标志性的特征就是他的全大写单词。Trump的竞争对手Joe Biden被Trump称为'Sleepy Joe'，是因为Biden表现的非常健忘，其tweet虽然没有Trump那么明显的风格特征，但是依然会语出惊人。于是，就想到了如果总统辩论采用的是Twitter的形式的话，会是怎样的场景。

随着自然语言处理的进步，现在的模型不仅仅能够做到一些基本的对话功能，也能够通过不同的语料库针对不同的特点进行建模，使得对话能够拥有自己的灵魂与特征。同时，Twitter作为时下流行的社交软件，其用户发送的文本数量大，且具有自己的特色，很多流行语以及有趣的梗都是从Twitter的用户发送的文本中产生的。Twitter限定其用户发送的Tweet字数小于120字，因此也是短文本生成与判别的很好的语料库。

要做到让这两位个性鲜明的总统能够用他们的Twitter风格进行对话，实际上需要对两者的语料库分别建立模型，才能生成出有自己风格特色的tweet。在这个项目中，我们采用了BERT-Tiny模型，以两位总统的2020年1月至6月发过的tweet为语料库，采用RoBERTa的训练方式对原始模型继续预训练。训练完成后，用生成式的方法(seq2seq模型)产生tweet。

最终，训练完之后的模型能够捕获到两位总统的特征，并且也能够理解上下文信息，生成出符合逻辑以及主题的tweet。

# 2 相关工作

## 2.1 对话系统

对话系统大致可分为两种:

1. 任务导向型(task-oriented)对话系统
2. 非任务导向型(non-task-oriented)对话系统(也称为聊天机器人)

任务导向型的系统旨在帮助用户完成实际具体的任务，例如帮助用户找寻商品，预订酒店餐厅等。

非任务导向的对话系统与人类交互，提供合理的回复和娱乐消遣功能，通常情况下主要集中在开放的领域与人交谈。而在这个任务中，我们需要用到的是非任务导向的对话系统，其方法主要可以分为两种：

1. 生成方法，例如序列到序列模型（seq2seq），在对话过程中产生合适的回复，生成型聊天机器人目前是研究界的一个热点，和检索型聊天机器人不同的是，它可以生成一种全新的回复，因此相对更为灵活，但它也有自身的缺点，比如有时候会出现语法错误，或者生成一些没有意义的回复；
2. 基于检索的方法，从事先定义好的索引中进行搜索，学习从当前对话中选择回复。检索型方法的缺点在于它过于依赖数据质量，如果选用的数据质量欠佳，那就很有可能前功尽弃。

因为BERT本身是一个seq2seq的模型，采用生成方法是更加自然的选择。

## 2.2 BERT

BERT的全称是Bidirectional Encoder Representation from Transformers，即双向Transformer的Encoder。模型的主要创新点是self-attention机制以及pre-train的方法，即用了Masked LM和Next Sentence Prediction两种方法分别捕捉词语和句子级别的表示。

### 2.2.1 模型结构

Transformer的结构本质是由一个encoder和一个decoder组成的，其中的注意力机制是Transformer好用的原因。

encoder层由一层多头注意力和一层全连接的前向传播层组成，该多头注意力层就是BERT创新的self-attention机制。

decode层由一层masked多头注意力层，一层多头注意力层和一层全连接的前向传播层组成。这两层注意力层分别是self-attention和encoder-decoder注意力。

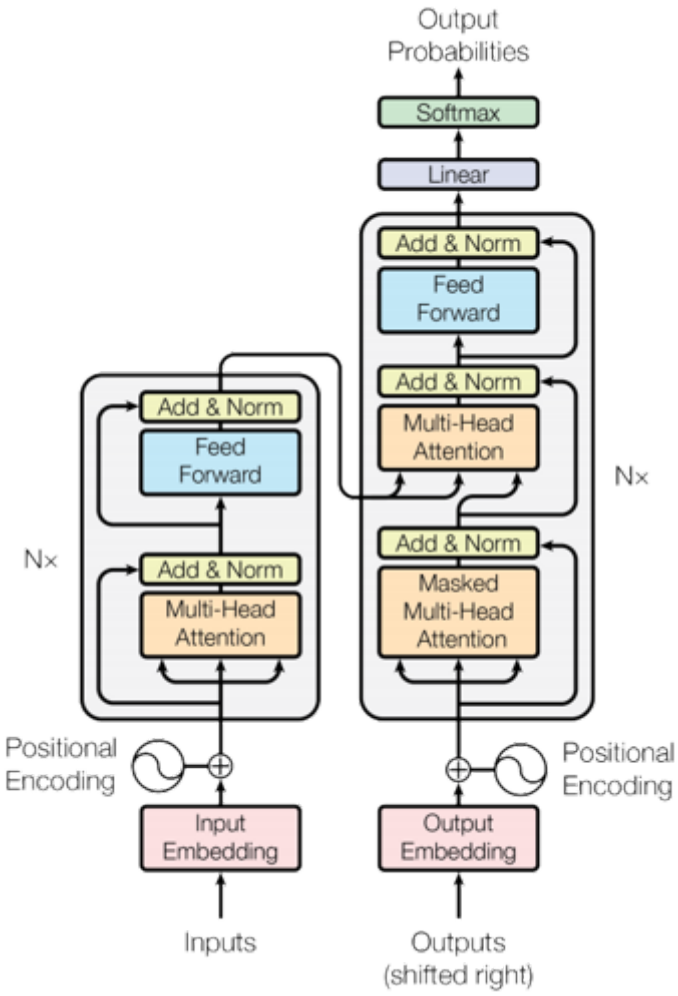


图1. Transformer结构

### 2.2.2 Attention机制

在论文中，BERT所使用的注意力机制被叫做“归一化的点积注意力”。在实现过程中，该注意力机制可以理解为将问询以及一组键值对映射起来，并最终输出注意力值，这些问询以及键值对都是由向量构成的。通常来说，习惯将问询组成一个矩阵$Q$，将键值对分别组成矩阵$K$和$V$，注意力的计算公式如下：

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

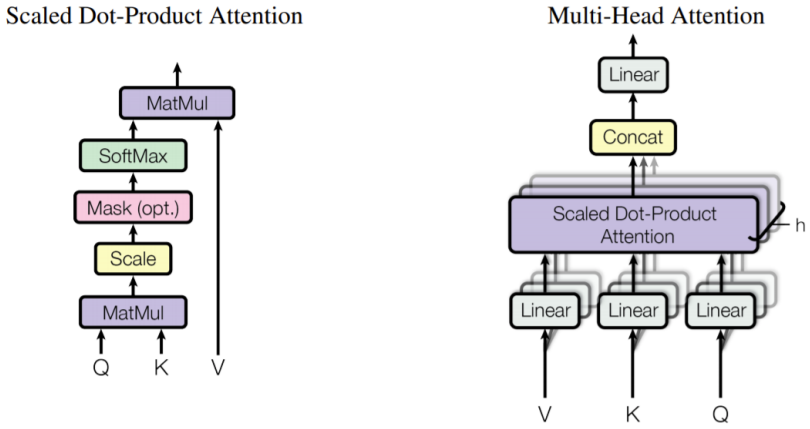


图2. 注意力机制

基于上述的注意力计算方法，BERT定义了多头注意力，并且运用在了两个不同的层面上：自注意力与encoder-decoder注意力

自注意力是指通过计算

### 2.2.3 词向量表示

BERT采用的输入向量有3种：词嵌入，段落嵌入以及位置嵌入。

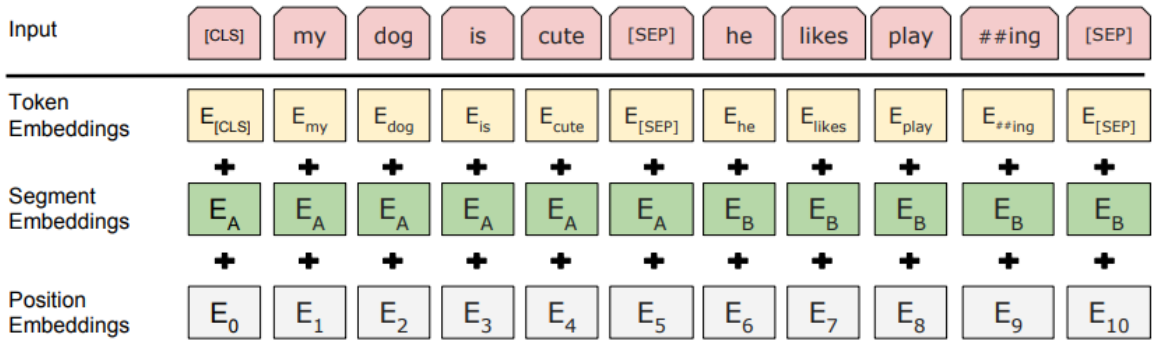


图3. 输入向量

词嵌入的第一个是[CLS]代表着开头，一句话的结尾是[SEP]。

段落嵌入是简单的标号，不同的句子标号不同，用来做上下文联系。

位置嵌入是利用三角函数的方式来学习的：

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_model})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_model})$$

### 2.2.4 预训练方法

BERT采用了两种预训练的方法：Masked Language Model以及Next Sentence Prediction。

MLM是指将随机的一个单词用[MASK]标识符代替，模型的任务就是预测这个[MASK]原来是什么单词。BERT对这个训练方法做的改进是当这个单词被选为mask单词时，80%的概率将这个单词替换为[MASK]，10%的概率将这个单词随机替换成其他单词，10%的概率不对这个单词做任何改动。这样的mask时候的trick能够帮助减少[MASK]这个标识符对模型的影响，同时，也能减少因mask使得单词没有出现过导致词嵌入没有被训练的概率。

NSP是指输入两个句子，用前句预测后句是哪一个，这种训练的方法能够使得模型理解上下文之间的关系。在这个训练方法中，语料库的选择就尤为关键，因为语料库中的上下文关系很大程度上决定了学习到的上下文之间的逻辑，因此，应该选用文档级别的语料进行训练。然而，在这个任务中，tweet构成的语料库都是以少于120字的短文本为主，没有办法学习到很多有关的上下文逻辑，采取的办法就是在原有的模型上继续预训练，从而保有之前学习到的上下文逻辑，这样才能够实现对话。

# 3 实现

该任务是基于bert4keras实现的，具体名词的解释等等可以查看链接中的文档。整体的框架采用的是tensorflow.keras，

## 3.1 数据处理

### 3.1.1 数据集

爬取这个任务所需要的数据集的最好的方法是通过Twitter的API，但是不幸的是，Trump的账号以及数据因为最近的白宫“起义”的事件被封禁了，同时，Twitter也拒绝了个人的API访问权限的申请，无奈之下只能从别的地方搜集数据。不幸中的万幸是Kaggle提供了详细的数据，从2009年到2020年6月的所有Trump以及Biden发过的tweet都有记录，因此采用了该数据集。数据集链接：Joe Biden， Donald

Trump。

因为Trump先生的tweet数量实在过于庞大，平均每天超过12条的数量，所以不得不对tweet的时间范围进行缩小，最后选取的tweet为2020年1月1日开始的所有tweet，两人各自都拥有约2000条左右的tweet作为语料库，对于短文本来说，这样的数量应该是足够了的。

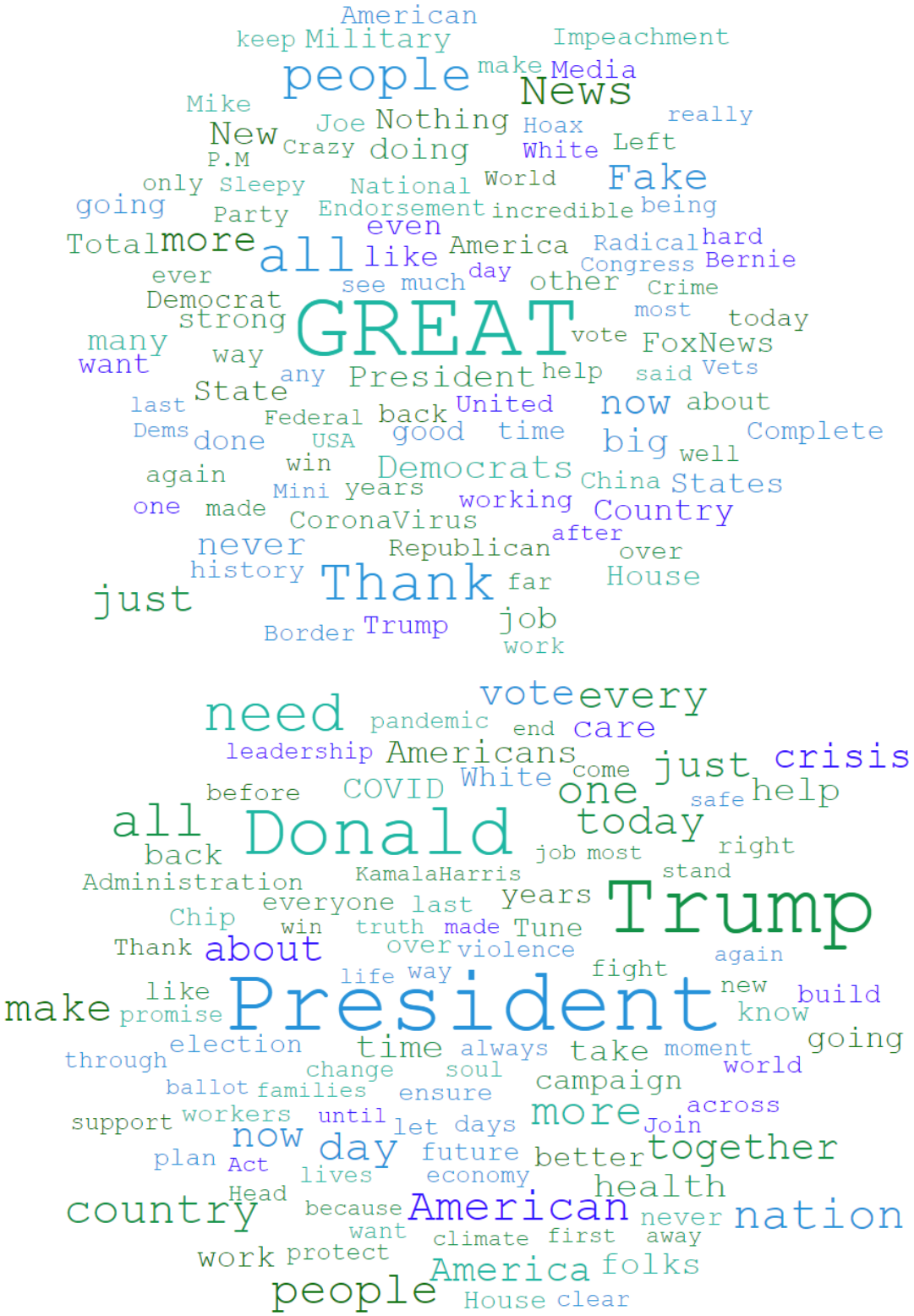


图4. Trump与Biden的词云

从上图的词云中，我们可以看出Trump的tweet中词频最高的词为GREAT，与其竞选的口号 'Make America GREAT Again' 对应了起来。当然，还有他在Twitter中经常发的 'Fake News' 以及 'Media'。而Biden这边就显得比较官僚且与原有的印象不同，似乎更加具有攻击性。他发的tweet中词频最高的词是president，可以看出其野心，同时，对于Donald Trump的点名还是很多的，几乎与president一样高，可以看出其攻击性还是很强的。

还可以对比两个人所用的词汇的长短，可以明显看出，Trump使用的词汇字母数少，这也是为什么Trump的tweet具有代表性的原因之一。

### 3.1.2 数据读入

从语料库转变为模型所需要的输入，还需要进行数据的处理。如上文所说的，BERT的输入由3种嵌入构成。词嵌入可以通过字典来表示，通过从字典中查找该单词对应的序号，作为词嵌入的值，该字典使用的是BERT-Tiny本身所使用的字典。段落嵌入是通过分句实现的，tweet中基本都是简单句，因此采用遇到标点符号开始分句的方法。位置嵌入则采用BERT原本的学习方法，利用三角函数进行学习。最终将处理好的数据以`tfrecord`的形式存入文件，便于之后的训练学习过程。

## 3.2 模型实现

### 3.2.1 模型结构

模型的结构是基于BERT-Tiny来实现的，词向量的大小为128，由两层Transformer组成，总的训练参数为约442万个。其中，词袋大小为30522，dropout概率为0.1，多头注意力个数为2，注意力dropout概率为0.1。整体的模型结构如下：

```
Attention-UniLM-Mask (Lambda)   (None, 1, None, None 0           Input-Segment[0][0]
__________________________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 128)    66048       Embedding-Dropout[0][0]
                                                                 Embedding-Dropout[0][0]
                                                                 Embedding-Dropout[0][0]
                                                                 Attention-UniLM-Mask[0][0]
__________________________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 128)    0           Transformer-0-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 128)    0           Embedding-Dropout[0][0]
                                                                 Transformer-0-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-0-MultiHeadSelfAtte (None, None, 128)    256         Transformer-0-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-0-FeedForward (Feed (None, None, 128)    131712      Transformer-0-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-0-FeedForward-Dropo (None, None, 128)    0           Transformer-0-FeedForward[0][0]
__________________________________________________________________________________________________
Transformer-0-FeedForward-Add ( (None, None, 128)    0           Transformer-0-MultiHeadSelfAttent
                                                                 Transformer-0-FeedForward-Dropout
__________________________________________________________________________________________________
Transformer-0-FeedForward-Norm  (None, None, 128)    256         Transformer-0-FeedForward-Add[0][
__________________________________________________________________________________________________
Transformer-1-MultiHeadSelfAtte (None, None, 128)    66048       Transformer-0-FeedForward-Norm[0]
                                                                 Transformer-0-FeedForward-Norm[0]
                                                                 Transformer-0-FeedForward-Norm[0]
                                                                 Attention-UniLM-Mask[0][0]
__________________________________________________________________________________________________
Transformer-1-MultiHeadSelfAtte (None, None, 128)    0           Transformer-1-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-1-MultiHeadSelfAtte (None, None, 128)    0           Transformer-0-FeedForward-Norm[0]
                                                                 Transformer-1-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-1-MultiHeadSelfAtte (None, None, 128)    256         Transformer-1-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-1-FeedForward (Feed (None, None, 128)    131712      Transformer-1-MultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-1-FeedForward-Dropo (None, None, 128)    0           Transformer-1-FeedForward[0][0]
__________________________________________________________________________________________________
Transformer-1-FeedForward-Add ( (None, None, 128)    0           Transformer-1-MultiHeadSelfAttent
                                                                 Transformer-1-FeedForward-Dropout
__________________________________________________________________________________________________
Transformer-1-FeedForward-Norm  (None, None, 128)    256         Transformer-1-FeedForward-Add[0][
__________________________________________________________________________________________________
MLM-Dense (Dense)               (None, None, 128)    16512       Transformer-1-FeedForward-Norm[0]
__________________________________________________________________________________________________
MLM-Norm (LayerNormalization)   (None, None, 128)    256         MLM-Dense[0][0]
__________________________________________________________________________________________________
MLM-Bias (BiasAdd)              (None, None, 30522)  30522       Embedding-Token[1][0]
__________________________________________________________________________________________________
MLM-Activation (Activation)     (None, None, 30522)  0           MLM-Bias[0][0]
==================================================================================================
Total params: 4,416,698
Trainable params: 4,416,698
Non-trainable params: 0
__________________________________________________________________________________________________
```

图5. 模型结构

模型结构的加载可以使用bert4keras中的`build_transformer_model`完成，它能够根据BERT模型的配置文件自动加载模型结构，因为我们需要基于原有的检查点继续预训练过程，所以模型的结构就不再做改动。

### 3.2.2 预训练

让BERT模型去学习一整个语料库的特征，从而改变其中的词向量的过程被称为预训练；而根据下游任务通过监督学习针对任务对模型中的某些参数进行调整的过程被称为微调。在这个任务中，没有明确的下游任务，而整体的思路是学习Trump与Biden的tweet的特征，因此需要进行预训练。但是，预训练相较于微调最大的坏处就在于时间代价过于高了，因为相较于调整全连接的某些参数，预训练需要调整所有的参数，其调整的参数量甚至要大于很多更大的BERT模型。因此，基于时间的考虑，只能采用BERT-Tiny模型。

预训练的方法与BERT论文中阐述的两种方法一致，采用MLM以及NSP。其中，MLM的概率采用RoBERTa论文中采用的参数，即15%的概率mask掉一个单词，80%的概率用[MASK]替换，10%用随机单词替换，10%不做任何的替换。

预训练使用的优化器为Adam，其参数参考了RoBERTa论文中使用的参数进行训练，具体参数的设置可以查看RoBERTa的论文。

最终，预训练两个模型在一张GTX1080Ti上花费了大约20小时，MLM任务的准确率达到了99%以上。

### 3.2.3 生成模型

因为时间的关系，没有尝试SOTA的seq2seq模型，采用了最简单的根据前文的信息预测下一个最有可能的单词是什么。即对之前的所有文本进行编码，利用这些编码预测下一个最有可能出现的单词。当然，这样的简单粗暴的方法会有许多问题，例如会出现大量重复的单词。在实现的时候，对下一个预测的词做了限制：不能出现在一定长度内的文本中，否则寻找下一个最有可能出现的单词。这样的模型随着上下文不断的增加，会使得运行时间不断增加。

# 4 实验结果

整体来说，对于tweet中频繁出现的主题，TrumpAI和BidenAI还是能够较好的理解的，并且一些特征也能够体现出来。同时，随着上下文的增加，我们可以看出模型对主题的理解越来越好。我们可以从3个表现比较好的例子中来分析。

## 4.1 Sleepy Joe

第一个例子是Trump对Biden的戏称 'Sleepy Joe'。

```
context: Sleepy Joe

Trump: You!
Biden: They wake up and tell you to say , folks were keeping these in what they
are doing on.

Trump: Thank you to keep up.
Biden: And where they are doing work workers.

Trump: Businesses are they family, jobs.
Biden: And where they are all in the few workers who have a long over those with
folks across what we're doing on.

Trump: They are doing.
Biden: But jack her h 9 md to the few folks who are with some of those in
workers're doing so many hardest we all more years.

Trump: We are our great country and they finally to keep people that greatest
farmers?
Biden: Two kings who are among the last of our nation ?

Trump: We are finally.
Biden: So much hardest than who are all of the people we re not have to any more
our greatest heroes that truly corrupt utterly defeated nation?

Trump: They are finally our military.
Biden: Not the generals of our military and decisive heroes who defeated us
fighters faced this country.

Trump: Our military bases have no 25 those on all americans .
```

```
Biden: The military and our generals who defeated decisive action that corrupt
heroes: incapable of doing this crisis.

Trump: Our military deployment s that are finally deployed to achieve freedom .
Biden: All military generals and decisive action defeated the us 3 allies that
our nation ?

Trump: our military and bases have most important countries , we are finally had
that these successes .
Biden: all of decisive action that we defeated the military and generals : our
allies , not heroic allied heroes are among incredible protective equipment who
save this crisis risking their own threats battlefield combat fighters fighting
so exceptional front lines forces for deter iranian essential shields .
```

从Biden的第一个tweet我们可以看出一开始模型只是学习到sleepy和Joe这两个分开的单词，其实并不知道Sleepy Joe指的是什么，但是很诡异的是Trump的tweet是'you !'。这是一条很有Trump个人风格的tweet，短小且带有很强的情感，并且对象直指Biden，不过感觉是蒙的（狗头）。第二条Trump的tweet也是很诡异，从人的角度来看似乎有点嘲讽Biden没看懂Sleepy Joe的含义，比较阴阳怪气，倒也确实符合Trump搞怪的这么一个角色设定。

其中不得不提到的是当Biden提到'Two kings who are among the last of our nation ?'，Trump的回答竟然是'We are finally.'确实很Trump。

后来从Biden理解sleepy为工作开始，画风逐渐跑偏，从商业聊到军事。从military这样一个很标志性的词开始，话题就固定了，后面他们针对军事开始发推。

## 4.2 Conora Virus

第二个例子是最近的新冠病毒，相信两位总统对这个问题的tweet还是比较多的。

```
context: conora virus

Trump: Is lid news business director ?
Biden: Take responsibility and his job safety insurance care.

Trump: Your great job !
Biden: His job is now .

Trump: He is strong on crime .
Biden: He is about his job .

Trump: He is a great job !
Biden: And his job is about safety .

Trump: He is doing everything that smoke and look at the fake news media says
talking flu ine dem 19 : 54 % .
Biden: Job got his lies about everything else is that he 1 ua a lot over safety
measures counts but is wrong ?

Trump: What he is the terrible crime .
Biden: But that is about his job in chief strategy .

Trump: He is the fake news media says talking nowhere happens again .
Biden: But he is about his job ?

Trump: He is the nowhere , and jeff dem ete facts nothing democrats talking
stories about leaking media leaked?
```

```
Biden: No claims filed unemployment debt tax of : safety measures is cost about
millions .

Trump: Money and now he knew the wa cky “ no matter who knows what everything
else nowhere , bernie dem facts nothing democrats talking ice house .
Biden: Countless lies over the truth is about wrong ?

Trump: He is saying the transcript evidence that it ’ s talking nowhere and media
wants “ russia .
Biden: No one is about the truth lies .
```

一开始几句就比较精彩（除去语法问题以外）,Biden的tweet是让Trump负起责任，并指明其职责是安全保障。然而一开始TrumpAI似乎没能很好的理解。直到第五句开始，积累了足够的上下文后，Trump面对新冠病毒时的标志性'fake news'出现了，我认为从这里开始TrumpAI可能开始理解了这个主题。

如果说4.1中的例子是和平讨论的话，这个例子中似乎有一点互相攻击的意思了，Trump的tweet中提到的'democrats talking stories about leaking media'以及Biden的'Countless lies over the truth is about wrong ?'还是比较精彩的。

## 4.3 Everyone should vote!

第三个例子是有关美国大选的，也应该是在tweet中出现频率较高的主题了。

```
context: Everyone should vote!

Trump: god bless you !
Biden: thee , thank you proud promise your support them for hope .

Trump: thank you to those will never be stronger .
Biden: i will proud to thank you , and honored as a support for your promise :
grace .

Trump: thank you to your vet for congress !
Biden: i promise you , thank your support them proud to give me out .

Trump: thank you to receive .
Biden: thank you proud to support your promise , i want for me .
```

一提到vote两者的反应就很雷同了，基本上都是以感谢为主，并且从第一轮发推就开始理解了主题，说明vote这个主题在他们的tweet中应该是出现的比较多，因此模型能够很好的预测。

## 4.4 结论与改进

总的来说，两个模型对于新语料库的加入的拟合还是比较可以的。Trump用词短、句子短甚至是感叹号的特征的抓取做的比较到位；Biden的tweet的那种死板的官僚作风也能够体现；并且，两个模型对于上下文的理解随着上下文长度的增加是越来越好的，最后都能针对同一话题进行发推。整体来说，对于语料库中频繁出现的话题，两个AI之间的对话也还能算一场小型的Twitter辩论。

当然不足的地方也有很多，从数据集的清洗不到位，到模型的深度，再到生成模型的选择，都有更好的选择，以下可能是最希望改进的地方吧。

1. 大小写区分

   这次使用的模型是对大小写不敏感模型，因为考虑到机器的性能，考虑大小写会对词袋的大小有着很大的影响，没有使用大小写敏感的模型。但是Trump发的tweet最明显的特征就在于他的大写字母的使用，例如'FAKE NEWS!'，失去了大小写对于Trump的个人风格的模仿还是欠缺了一些，同时从生成的语句来说，可读性稍差了一点。

2. 数据集及模型大小

   这次数据集没有做很多清洗，导致一些网页链接仍旧混在数据集中。这也是自己的疏忽，一开始看了几个tweet比较正常就没有再看，到后来生成词云的时候才发现https是一个出现频率还挺高的词。可惜没有时间再去训练一遍了。这些无用的数据影响了tweet的生成，导致会出些一些类似乱码的单词。

   同时，选用的数据集以及模型都偏小，如果有时间，可以采用更加大的数据集和模型，效果肯定会更加好一些。

3. seq2seq模型的选择

   生成模型上没有花太多的心思，也没有查找一些论文来看一下，完全是跟着感觉走。可能采用更加好的生成模型以及方法对于句子逻辑以及语法会有比较大的提升。

# 5. 感想

这是第一次从头至尾体验完成一个NLP项目，从确定题目到数据集的获取，再到模型的选择，代码的实现和等待训练的煎熬，以及最后完成时看着人工智障之间的对话，还是挺开心的，也花了几个小时尝试不同的主题以及他们之间的对话，看到一点有趣的地方也是挺欣慰的。当然，也体会到了深度学习是极度依靠硬件的，硬件的好坏决定了学习训练的时间，也就决定了一个模型的大小。最后从这个项目，体会到一个NLP项目的开发过程以及其中的重点和难点。

# 参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need"

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach"

[3] Lifeng Shang, Zhengdong Lu, Hang Li, "Neural Responding Machine for Short-Text Conversation"