

# 语言分析与机器翻译报告

李明隆 2001765 计硕 2004

## 摘要

本报告主要介绍项目中三种对联自动生成的方法，三种方法分别为隐马尔可夫模型（HMM）、最大熵马尔可夫模型和 LSTM 模型。本文将在三节中分别对这三种方法进行介绍，对项目运行的过程、结果进行展示并对相应的部分进行解释。

## 1. 介绍

机器翻译，又称为自动翻译，是利用计算机将一种自然语言(源语言)转换为另一种自然语言(目标语言)的过程。它是计算语言学的一个分支，是人工智能的终极目标之一，具有重要的科学研究价值。同时，机器翻译又具有重要的实用价值。随着经济全球化及互联网的飞速发展，机器翻译技术在促进政治、经济、文化交流等方面起到越来越重要的作用。机器翻译技术的发展一直与计算机技术、信息论、语言学等学科的发展紧密相随。从早期的词典匹配，到词典结合语言学专家知识的规则翻译，再到基于语料库的统计机器翻译，随着计算机计算能力的提升和多语言信息的爆发式增长，机器翻译技术逐渐走出象牙塔，开始为普通用户提供实时便捷的翻译服务。

机器翻译发展到现在产生了许多框架，下面进行简单介绍。基于规则的机器翻译：早期的机器翻译研究都是以基于规则的方法为主，它的主要思想是以词典和人工书写的规则库作为翻译知识，用一系列规则的组合完成翻译。针对基于规则的方法存在的问题，基于实例的机器翻译被提出。该方法的基本思想是在双语句库中找到与待翻译句子相似的实例，之后对实例的译文进行修改，如替换、增加、删除等一系列操作，从而得到最终译文。统计机器翻译：它利用统计模型从单/双语语料中自动学习翻译知识。具体来说，可以使用单语语料学习语言模型，使用双语平行语料学习翻译模型，并使用这些统计模型完成对翻译过程的建模。整个过程不需要人工编写规则，也不需要从实例中构建翻译模板。神经机器翻译：在神经机器翻译中，词串被表示成实数向量，即分布式向量表示。这样，翻译过程并不是在离散化的单词和短语上进行，而是在实数向量空间上计算，因此它对词序列表示的方式产生了本质的改变。通常，机器翻译可以被看作一个序列到另一个序列的转化。在神经机器翻译中，序列到序列的转化过程可以由编码器-解码器框架实现。其中，编码器把源语言序列进行编码，并提取源语言中信息进行分布式表示，之后解码器再把这种信息转换为另一种语言的表达。

接下来的三节中将对项目中采用三种机器翻译方法来生成对联进行介绍并在最后一节给出对比分析。

## 2. HMM

### 1.) HMM 介绍

隐马尔可夫模型 (Hidden Markov Model, HMM) 是经典的机器学习模型, 它在语音识别、自然语言处理等领域得到了非常广泛的应用。其本质是一个概率模型, 用来描述一个含有隐含参数的马尔可夫过程, 简单来说, 是用来描述一个系统, 它隐含状态的转移和可见状态的概率。

一般来说, HMM 包含下面三个问题:

1. 估计: 即给定模型, 根据可见状态链, 计算在该模型下得到这个结果的概率, 这个问题的解决需要用到前后向算法。
2. 参数学习: 即给定隐含状态数量, 根据多个可见状态链, 估计模型的参数, 同 IBM 模型的参数训练一样, 这个问题的求解需要用到 EM 算法。
3. 解码问题: 即给定模型和可见状态链, 计算在可见状态链的情况下, 最可能出现的对应的状态序列, 这个问题的求解需要用到基于动态规划方法, 在 HMM 中被称作维特比算法 (Viterbi Algorithm)。

### 2.) 项目

项目的实验环境在里介绍, 后面两种方法的环境相同。

项目使用 Python3.7, TensorFlow1.50, 数据集下载地址在项目文件夹下 README 文件中说明。

项目文件夹中, model\_HMM.py、train\_HMM.py 和 demo\_HMM.py 三个文件分别对应 HMM 的模型、训练和使用。

图 1 是模型训练和根据几个对联的上联自动生成下联的结果。

得到的结果为:

上联 (输入)	下联 (结果)
重阳节需要登高山	明月心不须凭大地
桃李争春	梅花竞秀
暖春天地宽	春秋水天下
烟锁池塘柳	风流水岸花
蚕为天下虫	我是人间草
白水兮高高	青山秀美好

从实验结果上来看, HMM 模型得到的下联还是不错的。

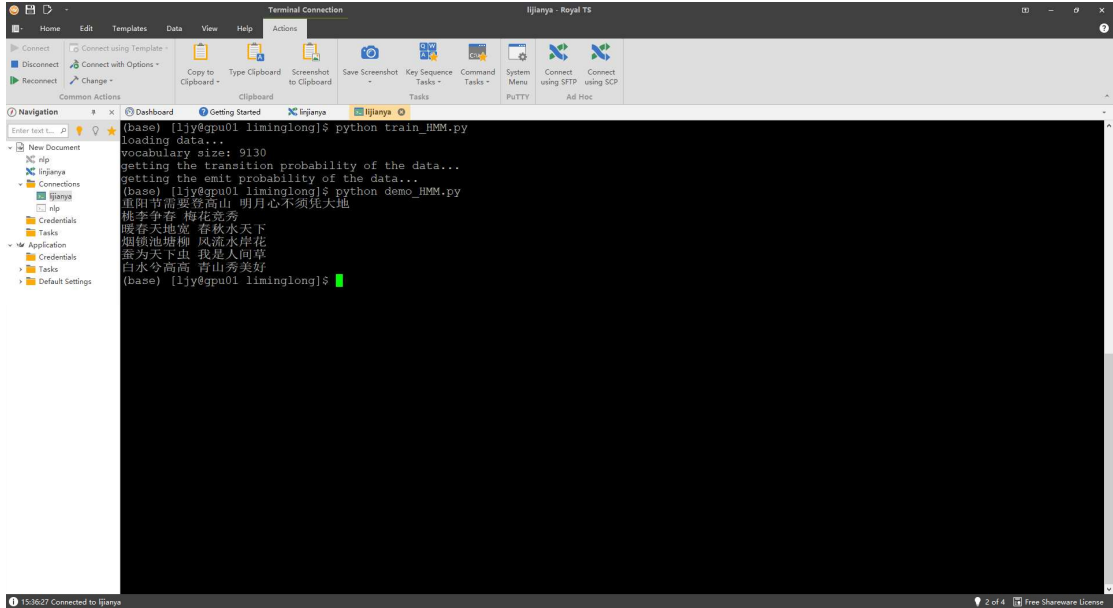


图 1: HMM 模型训练和结果

## 4. MEMM

### 1.) MEMM 介绍

最大熵马尔科夫模型 (MEMM) 利用判别式模型的特点, 直接对每一个时刻的状态建立一个分类器, 然后将所有的分类器的概率值连乘起来。为了实现是对整个序列进行的分类, 在每个时刻 $t$ 时, 它的特征不仅来自当前观测值 $x_t$ , 而且还来自前一状态值 $y_{t-1}$ 。所以 MEMM 中, 给定观测序列 $x_1^n$ , 某个状态序列 $y_1^n$ 的概率是:

$$P(y_1^n | x_1^n) = \prod_{t=1}^n P(y_t | y_{t-1}, x_t)$$

其中对于前一时刻可能的状态取值 $y_{t-1} = y'$ 和当前观测值 $x_t$ , 当前状态取值 $y_t = y^*$ 的概率通过最大熵分类器建模为:

$$P(y_t = y^* | y_{t-1} = y', x_t) = \frac{1}{Z(x_t, y')} \exp \left( \sum_a \lambda_a f_a(x_t, y', y^*) \right)$$

其中 $a, \lambda_a, f_a$ 分别表示特征函数数量, 特征函数权重和第 $a$ 个特征函数,  $Z(x_t, y')$ 表示归一化因子为:  $Z(x_t, y') = \sum_y \exp(\sum_a \lambda_a f_a(x_t, y', y))$ , 其中 $y$ 表示所有的可能状态取值。

之后使用维特比算法进行解码。

### 2.) 项目

项目文件夹中, `model_MEMM.py`, `train_MEMM.py` 和 `demo_MEMM.py` 三个

文件分别对应 MEMM 的模型、训练和使用。

图 2 是模型训练和根据几个对联的上联自动生成下联的结果。

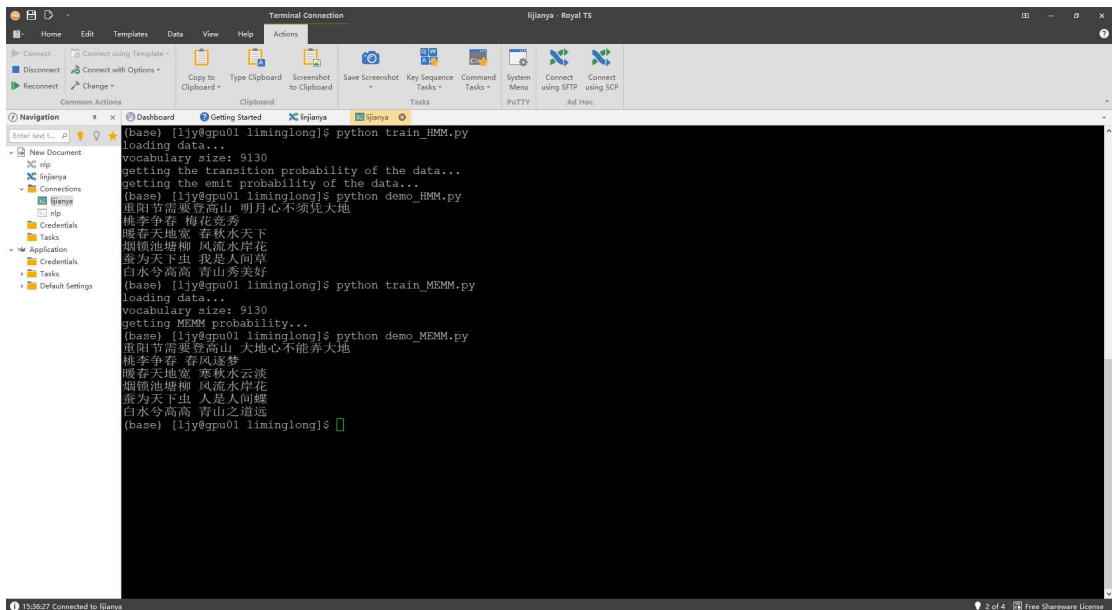


图 2: MEMM 模型训练和结果

得到的结果为:

上联（输入）	下联（结果）
重阳节需要登高山	大地之心不能弄大地
桃李争春	春风逐梦
暖春天地宽	寒秋水云淡
烟锁池塘柳	风流水岸花
蚕为天下虫	人是人间蝶
白水兮高高	青山之道远

从得到的下联来看，MEMM 在小部分上联得到的结果并不是很理想，但是大部分的结果还是较优的。

## 4.LSTM

### 1.) LSTM 介绍

循环神经网络的核心是设计循环单元的结构。至今，研究人员已经提出了很多优秀的循环单元结构，这里将介绍其中三种基本结构：RNN，LSTM 和 GRU。LSTM 和 GRU 是 RNN 的变体，在自然语言处理任务中得到了广泛的应用。RNN 结构使得当前时刻循环单元的状态包含了之前时间步的状态信息。但是这种对历史信息的记忆并不是无损的，随着序列变长，RNN 的记忆信息的损失越来越严重。在很多长序列处理任务中（如长文本生成）都观测到了类似现象。对于这个问题，Hochreiter 和 Schmidhuber 提出了长短时记忆（Long Shortterm Memory）模型，也就是常说的 LSTM 模型。

LSTM 的结构主要分为三个部分：

1. 遗忘。顾名思义，遗忘的目的是忘记一些历史，在 LSTM 中通过遗忘门

实现。

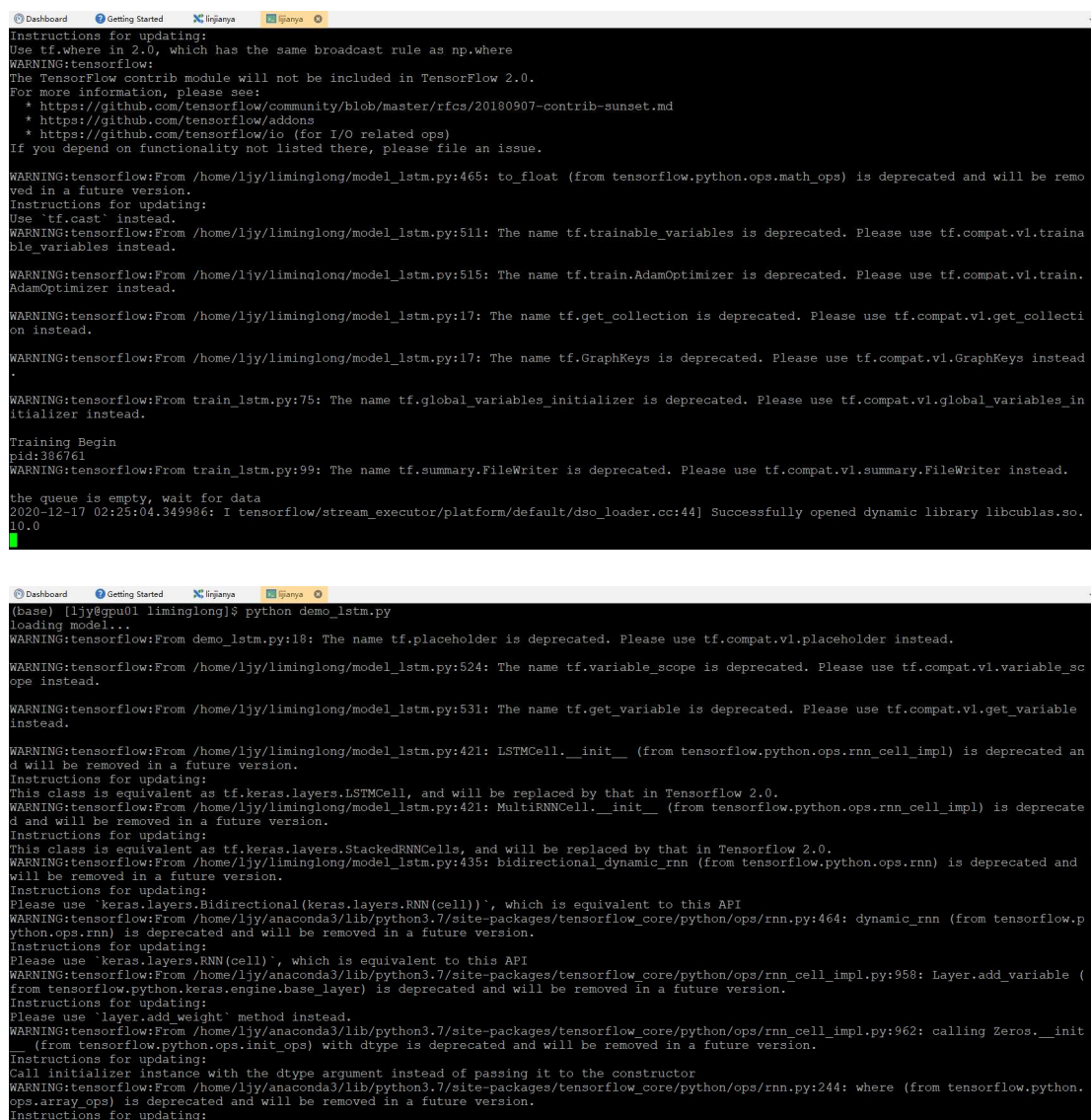
2. 记忆更新。生成当前时刻需要新增加的信息，该部分由输入门完成。
3. 输出。该部分使用输出门计算最终的输出信息。

## 2.) 项目

对联生成，即能够根据输入的上联，生成与之匹配的下联。春节、结婚、寿诞、乔迁、丧事等都用到对联。对联非常符合序列到序列问题的定义。比如，只需要把 整理好的对联数据（上下联对应）送给神经机器翻译系统，系统就可以学习上下联之间的对应关系。当用户输入新的上联，系统可以自动“翻译”出下联。

项目文件夹中，`model_lstm.py`、`train_lstm.py` 和 `demo_lstm.py` 三个文件分别对应 LSTM 的模型、训练和使用。

图 3 是模型训练和根据几个对联的上联自动生成下联的结果。



```
Dashboard Getting Started Injinya Injinya
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
* https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
* https://github.com/tensorflow/addons
* https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:465: to_float (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:511: The name tf.trainable_variables is deprecated. Please use tf.compat.v1.trainable_variables instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:515: The name tf.train.AdamOptimizer is deprecated. Please use tf.compat.v1.train.AdamOptimizer instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:17: The name tf.get_collection is deprecated. Please use tf.compat.v1.get_collection instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:17: The name tf.GraphKeys is deprecated. Please use tf.compat.v1.GraphKeys instead.
WARNING:tensorflow:From train_lstm.py:75: The name tf.global_variables_initializer is deprecated. Please use tf.compat.v1.global_variables_initializer instead.

Training Begin
pid:386761
WARNING:tensorflow:From train_lstm.py:99: The name tf.summary.FileWriter is deprecated. Please use tf.compat.v1.summary.FileWriter instead.

the queue is empty, wait for data
2020-12-17 02:25:04.349986: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0

(base) [ljy@gpu01 liminglong]$ python demo_lstm.py
loading model...
WARNING:tensorflow:From demo_lstm.py:18: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:524: The name tf.variable_scope is deprecated. Please use tf.compat.v1.variable_scope instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:531: The name tf.get_variable is deprecated. Please use tf.compat.v1.get_variable instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:421: LSTMCell.__init__ (from tensorflow.python.ops.rnn_cell_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This class is equivalent as tf.keras.layers.LSTMCell, and will be replaced by that in Tensorflow 2.0.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:421: MultiRNNCell.__init__ (from tensorflow.python.ops.rnn_cell_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This class is equivalent as tf.keras.layers.StackedRNNCells, and will be replaced by that in Tensorflow 2.0.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:435: bidirectional_dynamic_rnn (from tensorflow.python.ops.rnn) is deprecated and will be removed in a future version.
Instructions for updating:
Please use `keras.layers.Bidirectional(keras.layers.RNN(cell))`, which is equivalent to this API
WARNING:tensorflow:From /home/ljy/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/ops/rnn.py:464: dynamic_rnn (from tensorflow.python.ops.rnn) is deprecated and will be removed in a future version.
Please use `keras.layers.RNN(cell)`, which is equivalent to this API
WARNING:tensorflow:From /home/ljy/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/ops/rnn_cell_impl.py:958: Layer.add_variable (from tensorflow.python.keras.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From /home/ljy/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/ops/rnn_cell_impl.py:962: calling Zeros.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From /home/ljy/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/ops/rnn.py:244: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
```

```
2020-12-17 02:33:41.716686: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 9384 MB memory) -> physical GPU (device: 0, name: Tesla T4, pci bus id: 0000:b8:00.0, compute capability: 7.5)
2020-12-17 02:33:41.717372: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:1 with 14087 MB memory) -> physical GPU (device: 1, name: Tesla T4, pci bus id: 0000:8d:00.0, compute capability: 7.5)
2020-12-17 02:33:41.719050: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:2 with 14087 MB memory) -> physical GPU (device: 2, name: Tesla T4, pci bus id: 0000:8f:00.0, compute capability: 7.5)
2020-12-17 02:33:41.720758: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:3 with 14087 MB memory) -> physical GPU (device: 3, name: Tesla T4, pci bus id: 0000:b3:00.0, compute capability: 7.5)
2020-12-17 02:33:41.722382: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:4 with 14087 MB memory) -> physical GPU (device: 4, name: Tesla T4, pci bus id: 0000:b4:00.0, compute capability: 7.5)
2020-12-17 02:33:41.724026: I tensorflow/core/common runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:5 with 14087 MB memory) -> physical GPU (device: 5, name: Tesla T4, pci bus id: 0000:b6:00.0, compute capability: 7.5)
WARNING:tensorflow:From demo_lstm.py:32: The name tf.global_variables_initializer is deprecated. Please use tf.compat.v1.global_variables_initializer instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:319: The name tf.get_collection is deprecated. Please use tf.compat.v1.get_collection instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:319: The name tf.GraphKeys is deprecated. Please use tf.compat.v1.GraphKeys instead.
testing...
2020-12-17 02:33:51.974030: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
重阳节需要登高山 明月夜不能逛故乡
桃李争春 教育竞辉
暖春天地宽 习古古今通
烟锁池塘柳 枫燃镇地汀
蚕为天下虫 燕乃山中王
白水兮高高 黑土也肥瘦
青山不墨千秋画 绿水无弦万古琴
两岸凉生蕉叶雨 一江春涨棉花风
无边落木萧萧下 不尽流云处处游
两只黄鹂鸣翠柳 一弯新月钓清溪
深秋帘幕千家雨 仲夏麦田万亩稻
月透柳帘窥案卷 风吹竹管动窗纱
(base) [ljy@gpu01 liminglong]$
```

图三：LSTM 模型训练和结果

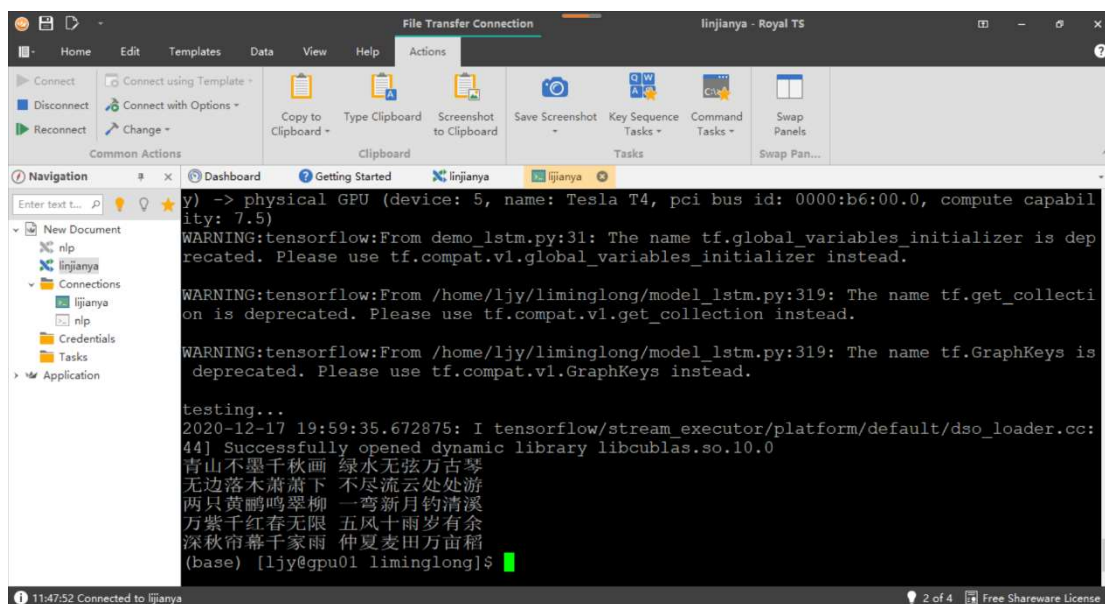
得到的结果为：

上联（输入）	下联（结果）
重阳节需要登高山	明月夜不能逛故乡
桃李争春	教育竞辉
暖春天地宽	习古古今通
烟锁池塘柳	枫燃镇地汀
蚕为天下虫	燕乃山中王
白水兮高高	黑土也肥瘦

## 5. 总结

```
linjiana - Royal TS
File Transfer Connection
Connect, Disconnect, Reconnect, Connect using Template, Connect with Options, Change, Copy to Clipboard, Type Clipboard, Screenshot to Clipboard, Save Screenshot, Key Sequence Tasks, Command Tasks, Swap Panels, Swap Panels...
Navigation
Enter text t...
New Document
nlp
linjiana
Connections
linjiana
nlp
Credentials
Tasks
Application
(base) [ljy@gpu01 liminglong]$ python demo_HMM.py
青山不墨千秋画 碧水无情万里春
无边落木萧萧下 万里飞花点点头
两只黄鹂鸣翠柳 一方紫燕舞红梅
万紫千红春无限 千红万紫燕有余
深秋帘幕千家雨 静夜月光万里风
(base) [ljy@gpu01 liminglong]$ python demo_MEMM.py
青山不墨千秋画 碧水无弦万里春
无边落木萧萧下 不尽长江滚滚来
两只黄鹂鸣翠柳 一条白鹭上红梅
万紫千红春无限 五湖四海风有余
深秋帘幕千家雨 古月镜头万里风
(base) [ljy@gpu01 liminglong]$ python demo_lstm.py
loading model...
WARNING:tensorflow:From demo_lstm.py:17: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.
WARNING:tensorflow:From /home/ljy/liminglong/model_lstm.py:524: The name tf.variable_scope is deprecated. Please use tf.compat.v1.variable_scope instead.
```





图四：三种方法的对比运行结果

根据上面的运行结果对三种方法进行对比分析：

1. 上联：青山不墨千秋画  
HMM：碧水无情万里春  
MEMM：碧水无弦万里春  
LSTM：绿水无弦万古琴

通过这个对比,可以看出神经网络更具有诗意。神经网络的输出把绿水当作琴,和上联的把青山当作画有个非常巧妙的照应。细细品味,让人置身于一个有声有色的山水画当中,耐人寻味。

2. 上联：无边落木萧萧下  
HMM：万里飞花点点头  
MEMM：不尽长江滚滚来  
LSTM：不尽流云处处游

这里 MEME 竟然给出了原诗的下联,而神经网络能给出一个合理的且非原诗的下联,可见其泛化能力和强大的创作能力。

3. 上联：两只黄鹂鸣翠柳  
HMM：一方紫燕舞红梅  
MEMM：一群紫燕舞红梅  
LSTM：一弯新月钓清溪

这里三个模型都给出合理的下联,但是神经网络的下联更加的富有诗意。新月相比满月在于它是像弯钩一样,它的倒影在小溪中,如同想要从溪水中钓鱼一般,想象力非常丰富,而其他两个模型的输出就少了这般诗意。

4. 上联：万紫千红春无限  
HMM：千红万紫燕有余  
MEMM：五湖四海风有余  
LSTM：五风十雨岁有余

这里看出,神经网络的输出更具有祝福气息,更符合新春的氛围。五风十雨解释为:五天刮一次风,十天下一场雨,形容风调雨顺,就是祝福风调雨顺,收获满满,岁岁有余。可以说神经网络是对词语的意思理解得非常透彻了。

5. 上联     : 深秋帘幕千家雨  
HMM     : 静夜月光万里风  
MEMM   : 明月镜头万里风  
LSTM   : 仲夏麦田万亩稻

可以看出, HMM 和 MEMM 的输出可谓前后不搭, 而神经网络的输出, “麦田”与“稻”遥相呼应, “千家雨”与“万亩稻”也对应得非常好。

从上面几个对比分析可以看出, 绝大多数情况神经网络的下联是由于其他两种方法得到的下联, 其得到的结果更具有诗意。HMM 和 MEMM 在多数情况下能得到合理的下联, 和 LSTM 相比生硬了些。