Folie 1

简介。

欢迎来到我们的展示。

Hello everyone

Welcome to our presentation.

自我介绍：

小组30，

We are group 30

这里是我们的甘特图（项目流程图）Gantt grafik

Here is Gantt chart of this task

借口： 因为在6月我们还有其他的考试 所以

在 end of June 我们才开始shared task in Deep learning for NLP 2020

Anyway，我们的短期项目取得了阶段性的成果，至少比baseline 稍微好一点。

Because of our other exams in June, we just started the task at the 27th June

Anyway, our short-term project has achieved some results and learned some project experience

现在我们将具体介绍我们组的项目

Now we will introduce some detail in our task

Folie 2

今天的presentation将会分成3个部分进行，分别是task 的任务描述，model的搭建以及相关结构，以及我们最终的项目成果

Today's presentation will be divided into 3 parts,

namely task description, modeling, and our final results

Folie 3

The task deals with semantic similarity under adversarial attacks.

Semantic similarity 照着念

We deﬁne STS as a supervised regression task in which the semantic similarity of two pieces of text (typically sentences) should be determined.

However, the test datasets have been perturbed with so-called adversarial attacks, which are modiﬁcations to the input of a model that do not change the label or score.

Such as visual attacks

我们在这里放了一个实例，左边和右边分别是攻击程度不同的句子，他们之间文本相似度度为 0.84

We have put an example here, the left and right are the sentences with different degrees of visual attack, and the text similarity’s score between them is 0.84

Folie 4

接下来在此我们罗列了在任务中具体落实的方法：其中包括preprocessing和processing部分

Next, we list the specific implementation methods in the task:

including preprocessing and processing

在preprocessing部分中，我们对样本数据进行了一定程度的预处理。

其中，我们将文本转化为word vector的形式，在这个转化过程中，我们使用了word vector的数据库：xxxx。

由于存在adversarial attacks，在我们转化word vector的过程中， OOV的出现不可避免， 基于VIPER做了简单的逆向adversarial attacks。目的是减少OOV的出现，简化我们的模型回归。

In the preprocessing, firstly we deal with the sentence data.

On the one hand, we converted the string text into the form of word vector, by using the pre-trained word vectors database: wiki news..

On the other hand, due to adversarial attacks, too many OOV vectors will be generated during the conversion into word vectors

To reduce OOV and to simplify our model regression. we make simple reverse adversarial attacks based on VIPER.

在processing部分中，我们将MLP和LSTM进行了串联。LSTM在这里发挥了embedding的作用，目的是考虑word的上下文关系。具体的预测过程依然出现在MLP里。  
此外我们还对具体的模型参数进行了随机搜寻，目标是找到一组合适的超参数，基于此来获得一个回归过程中的合适的映射。

In processing, due to the consideration of the word’s context we firstly try to connect LSTM and MLP in series. LSTM plays the role of embedding here, and the output of LSTM is used as the input of MLP.

In addition, to reduce OOV and simplify our model regression, we also set a random search for model parameters.

Folie 5

现在我们进入模型部分

这张ppt展示了我们模型的工作流。我们将模型划分成三个子任务。分别是预处理，处理及优化，以及结果预测。

Now we will introduce the model part.

This slide shows the workflow of our model. We divide the model into three subtasks. These are pre-processing, processing and optimization, and result prediction.

在预处理过程中，我们读取了数据，并且对数据进行了adversarial attacks的逆向还原，之后将还原过的句子转化成word vector。

In the preprocessing, we load the dataset and reverse the attacked character, and then convert it into word vector.

在处理及优化过程中，我们对模型进行了10次random search，每次random search会随机搜寻40组不同的超参数，最终我们对比每次random search的最优参数，从中选取test score，（即mse）最低的参数作为最终超参数，并进行最终的结果预测。

By Modeling, we set 10 random searches,

In each search we try randomly 40 different sets of hyperparameters.

Finally, we compare them, select the best one from them, and use it to the final result prediction.

Folie 6

MLP的模型如右图所示，我们首先基于句子中各个单词的word vectors计算出了一个平均vector，将这个平均vector视作句子整体的vector，将其作为MLP的输入量。

因为每个data set需要对2个句子进行拟合，因此我们有两个input vector，通过merge层将其合并，并作为MLP的最终输入量。

为了避免过拟合的发生，在每个Dense层之间我们放入了一个dropout层。

针对MLP我们也进行了random search的优化，hyper parameter的优化结果在表格里呈现。

The MLP model is shown on the right. We first calculate an average vector of every sentence and set it as the input of the MLP.

Because each data set needs to fit 2 sentences, we merge 2 input vectors and used as the final input of the hidden layers

To avoid overfitting, we put a dropout layer between each Dense layer. For MLP, we have also optimized hyper parameters with random search.

Results are presented in the table.

Folie 7

在MLP模型的基础上，我们希望对其进行拓展，一个句子里的单词会存在上下文之间的关系，基于阅读顺序，这个关系对于单个句子而言，通常是顺序的，即从左至右。 我们希望模型把这种上下文的关联也考虑进去，以增加最终预测语义相似度的准确性。因此我们采用了LSTM模型作为embedding 部分，对句子中的单词进行预处理，其处理结果将作为MLP的input。

对于MLP部分我们也进行了一定的修改。为了增加层与层之间的收敛效果，我们在每个hidden layer里插入了BatchNormalization 层。其他部分和原始MLP模型保持类似。

右边可以看到我们模型经过random search之后的具体参数。

Based on MLP, we hope to expand it. The words in a sentence will have a relationship between contexts. Based on the reading order, this relationship is usually sequential for a single sentence, that is, from left to right.

the model will also take this context into account, to increase the accuracy of final prediction

Here we have considered context.

So we use the LSTM model as the embedding part, and the processing result will be used as the input of MLP.

We have also made some changes to the MLP part. In order to increase the convergence effect between layers, we insert a BatchNormalization layer in each hidden layer.

On the right, you can see the specific parameters of our model after random search.

Folie 8

我们现在进行结果复盘。

从CodaLab的结果可知，在为期两天的development阶段，LSTM做为embedding的效果并不好，因此在test阶段，我们续用了MLP作为预测模型。其预测的分值为29.6%。

We now take overview of the result.

From the results of CodaLab, we can see that in the two-day development phase, LSTM is not a good result for embedding, so in the test phase, we renewed MLP. In the end, its predicted score was 29.6%.

此外，我们通过观测10次random search的最佳结果，发现对于大部分MLP而言，dense hidden layer的层数小于3，即除了输出层以外最多包含2层。对于我们的模型而言，如果dense的层数过多，那么很容易导致overfitting的发生。

By observing the best results of 10 random searches, we found that the number of dense hidden layer is less than 3, that is, it contains at most 2 layers except the output layer.

For our model, if there are too many layers of dense, it is easy to cause overfitting.

通过处理OOV我们发现，预测的结果对oov的word vetocr敏感。我们发现inverse attacks的恢复要比搭建模型重要，如果能够重建未被攻击的预测语句，那么结果将异常精准。

最后的最后，时间管理很重要，一周不到的时间对于这个项目而言还是太短。

By processing OOV, we found that the predicted results are sensitive to oov's word vector.

We found that reverse attacks are more important than building models.

If we can reconstruct the un-perturbed sentences, the result will be extremely accurate.

In the end, time management is very important. Less than a week is too short for this project.