1. Hi everyone, we are group 8 and I am the presenter today, my name is xxx. My team members include XXX, and KAR CHUN TEONG who is currently in Malaysia joining us online.
2. We attempted 3 different model architectures for this project, they are listed on this slide.
3. First, let’s look at DMRNets. This architecture is analysed and implemented by KAR CHUN TEONG
4. What is DMRNets? It’s a variation of Deep convolutional neural networks with Merge-and-Run mappings, coincidentally they have the same short-form DMRNets, so to differentiate, we use DMRNets(CNN) and DMRNets(FCN) in the later slides to refer to them. DMRNets(CNN) itself is a variation of Residual Neural Network (ResNet).
5. To explain what is DMRNets, we need to start from ResNet, so what is ResNet? ResNet originated from CNN, researchers have been pondering at the question of, does stacking more convolutional layers (which is increasing the depth) leads to better performance? They found out that stacking layers leads to a degradation problem, as they increase the depth, accuracy get saturated then degrades rapidly, and this is not caused by overfitting, the more layers they stack, the higher the training error. Some proposed a solution which is residual mapping, as shown on the image, denote the desired underlying mapping as H(x) = F(x) + x, instead of directly fitting to H(x), fit the layer to F(x), and fit another layer to the identity mapping which is x, at the end merge them. Given that the presentation time is limited, anyone interested in the details, for example: why they did this and how this improve the performance is encouraged to read the original article.
6. Next, we move on to the variation based on ResNet, the DMRNets(CNN). Basically what it does is to assemble the residual branches in parallel through a merge-and run mapping. By Merge it means to average the inputs of these residual branches, and by Run it means to add the average to the output of each residual branch as the input of the subsequent residual branch.
7. To further illustrate the model, please look at these three building blocks.
   1. (a) shows 2 residual branches assembled sequentially.
   2. (b) shows 2 residual branches assembled in parallel and with identity mappings.
   3. and finally (c) shows 2 residual branches assembled in parallel and with the proposed merge-and-run mappings.

Again, anyone interested in the details is encouraged to read the original article.

1. At last, we come to the model we use in this project, the DMRNets(FCN). Actually the only variation we did is to change the convolutional layers to fully connected layers. The model structure is shown below, actually I think a picture could be a better description so let’s directly skip to the next slide.
2. Here’s part of the model structure, first we define a fc\_block which consists of dense layer, batch normalization layer, leaky ReLU layer, and dropout layer. Then we define the merge-and-run function, which separate input into 2 branches, branch (a) pass through the fc\_block while branch (b) pass through identity mapping, which is essentially mapping to itself, then we pass both branches into an average layer, then the output passes through another leaky ReLU layer. So in this picture we can see both input first go through a fc\_block, then into a merge-and-run mappings. The model is actually so deep that the rest of the structure is cropped out, but it’s essentially the same process, repeatedly pass through fc\_block and merge-and-run mappings and finally merged into a single output.
3. For unknown reasons, the training of the DMRNets model did not converge, we tried modifying various parameters and adjusting the depth of the model, but in the end it couldn't work.
4. Now we look at the second model we use, the LSTM model, this model is analysed and implemented by TAKEHIRO MATSUNAGA.
5. LSTM is a type of recurrent neural network (RNN) architecture. A common LSTM unit is composed of a cell and 3 gates. The cell stores values over time intervals as cell states, and the 3 gates’ main job is to regulate the flow of information into and out of the cell. Specifically, forget gate decides which information to forget, input gate decides to update which value of cell states and update cell states, and at last, the output gate output filtered cell states. To put it in an abstract high-level view, what LSTM do is using past context as reference to predict the future output. For example, trying to predict the next word based on given sentence.
6. As our brain can understand articles based on knowledge that we learned before, RNN works in a similar way, RNN can derive some parameters from its forward node, this was what traditional neural networks could not achieve. However, traditional RNN could not store what it learned from forward node for a dragged time period because of its algorithm. Then LSTM was invented to solve this problem
7. This picture shows a cell of LSTM, the forget gate will selectively forget some information getting from forward cell. And the formula determine which data should keep and which one should forget.
8. Performance.
9. The third model we use is bidirectional LSTM model (biLSTM). this model is Analysed and implemented by TAKEHIRO MATSUNAGA.
10. biLSTM is a variant of LSTM, while LSTM uses past context only, biLSTM uses both past and future context. biLSTM generally provided more context, which leads to better performance, for example, trying to predict the next word based on the given sentence, apparently using only one of the below sentences to predict the word is hard, but with both forward and backward sentences we could predict the word with more confidence.
11. This picture shows our model structure, actually its quite similar to the LSTM model, just change the layer from lstm to bilstm. Beside layer changing, I inspired by merging method and contribute the model to bilstm + merging method.
12. Performance
13. Experiment and analysis, this part is Tested by TAKEHIRO MATSUNAGA and Analysed by KAR CHUN TEONG
14. We wanted to use cloud platforms to train our model, but most of them are expensive, and we don't have the fund for that, Google Cloud Platform offers 1 month free trial, but to get the free trial we need to register with credit card, which unfortunately we don't possess, at the end of the day we need to train the model using our local machines. Limited by the performance of our local machines, we could only use a small subset of the datasets for training (the first 200 training shards and the first 100 validation shards).
15. This figure shows the graph of training curve of the LSTM model.
16. This figure shows the graph of training curve of the biLSTM model. As shown in both figures, both models converged at around $0.75$, with biLSTM having a slightly better performance.
17. The performances of 3 models are listed in the table, as we could see, the MAP@10 for the first 200 training shards and 100 validation shards are always higher than the MAP@10 for the whole datasets, the reason is obvious as we use the first 200 training shards and 100 validation shards as training datasets which leads to possible overfitting. However, we consider that the LSTM and biLSTM's performance of > 0.75 for the whole datasets as satisfactory, since we only use an extremely small subset to train them. We hypothesize that if we use larger datasets for training, the model would lead to more satisfactory results, too bad that we don't have the computing power to prove this hypothesis.
18. Contributions. (read the ppt or just let them read by saying “the contributions of each member are listed here”)
19. conclusion, Based on the experiments, we conclude that the biLSTM model has a superior performance over the other models we used. Therefore we use the biLSTM model in our final submission.
20. Thank you!