1.

Good morning all. I am Malavika and this is Christoffer sharing the screen, and the title of our project is search engine for movie cast generation. ***NEXT***

2.

The film industry in itself is a billion-dollar business which has a huge economical impact. The success of a movie depends highly on its cast. That is, its lead actors. ***NEXT***

3.

And for directors choosing actors for their movies, it is a challenging task given the huge number of actors and movies they make every year. Manually skimming through the humongous data to find relevant actors is nearly impossible. In order to assist them put together actors based on their requirements is what this project is about. ***NEXT***

4.

This leads us to our formal problem statement. The user is required to give three inputs, namely, the genres, the actor(s)' description such as gender and age interval along with a plot summary for the movie they are planning to make. Our system would provide a ranked list of relevant actors or group of actors (based on user input). ***NEXT***

5.

Here is a use case example. Let's say the user inputs are as follows. Genre being Adventure, Drama and Scifi. The user wants three actors, a male between age 45 and 55, two females between 30 to 40 and 40 to 50 respectively. The user has also given a short description of the movie plot

Our system takes these inputs and gives out a ranked group of actors as follows. The top ones are Matthew McConaughey, Anne Hathaway and Jessica Chastain with a group score of 9.1. Movie buffs would immediately recognise that this is the cast of the blockbuster Interstellar. We gave the input genres, actors' descriptions and plot of interstellar as It is from IMDb. ***NEXT***

******

11.

We also did the implementation in python to compare the performance between spark and pure python. The python workflow is essentially the same as that of spark. ***NEXT***

12.

Speaking of the challenges we faced during this project, these three tops the list. ***NEXT***

13.

The first one was the evaluation of plot similarity. After we generate a candidate actors' list based on their descriptions, we look for similarity between the input plot and plot summaries of the movies that these actors have acted in previously. For this purpose, we first tried the cosine similarity with tf-idf, which is well-known for detecting similar documents in text mining. But this gave us unsatisfactory results since, this model depends highly on the number of words in a document. Which is why this was a bad choice for our task, since our plot summaries are small, with very few words. We then tried the word2vec model which is a neural network as the same suggests converts words to vectors and computes various mathematical distances between them. This method was

highly promising but yielded poor results since it was computing word-wise similarity while we needed sentence-synonymity.

That is when we chose the universal sentence encoder tensorflow model by google. This is also a word2vec model but is instead trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. This perfectly matched our criteria and gave excellent results. This model would return a value between 0 and 1 which we multiplied by 10 to normalise it like the genre score. ***NEXT***

14.

The bigger problem with using this model on spark is its size which is nearly 1GB. It usually halts for 30 seconds before the import is done. Loading the model was equally expensive, thus the total time on each partition was a minute before it could start. Because there were multiple partitions per node, the model was loaded several times on each of them. Which is highly expensive and was unacceptable. We looked for various alternatives and ended up doing the calculation on the master node. This means that all of the data had to be transferred over the network to the master node, and then back again to the slave nodes which is not advisable but this was actually faster in our cluster than the former method. The runtime of similarity score is typically about 8 minutes. ***NEXT***

15.

Another major issue was the evaluation of results. We were posed with the question: How do we measure the quality our results? First off, we assume that any movie with a higher rating on IMDb has a good cast. That is, for a given genre, if out of two movies, one has a higher user rating, we assume that its actors are good performers in that genre and vice versa. So we decided to check our system by giving existing movie attributes from IMDb as input, if the original actors show up early in our results, it is considered a good output.

As shown in our use case example, when we gave attributes of the movie interstellar as input, we got the original actors with the highest score. Which is exactly what we want. 😊

******

22.

As we've seen the performance of a system with very large data and complex computations are exponentially improved by the use of big data tools like Spark and Hadoop. And thus is highly recommended by various other researchers. ***NEXT***

23.

Here are a few related works. ***NEXT***

THANK YOU.

If you have any questions, let us know.