<u>Project Overview</u>

<u>Background</u>

Luckin Coffee is a Chinese company which was listed on Nasdaq on May 16th, 2019. Early this year, Muddy Waters revealed a report which claimed that Luckin Coffee was inflating its number of items sold per store. After Lucking announced on April 2 that it would conduct an internal investigation into fraud allegations. Luckin's stock price dropped more than 80%. This event aroused our interests. For this project, we decided to use Luckin's earning announcements to develop a graphical database to apply the model to automated predictive text completion.

<u>Approach</u>

For our mid-term submission we focused creating a predictive text filler through python coding. For the final assignment submission, we made modifications to some of this code to allow it to be integrated into Neo4j to create a graph database.

<u>Methodology</u>

Data: Luckin Coffee Earnings Call Transcript was attained online and converted into a txt file for ingestion

NLTK: Python's built-in programs to work with language data. It provides us with sample texts and tutorial code.

N-gram model: the model that can assign probabilities to the sequences of words. It allows our program to output the predicted text.

Neo4j: the graph database software. It can help us to a build graph database which can be connected to python programs by API.

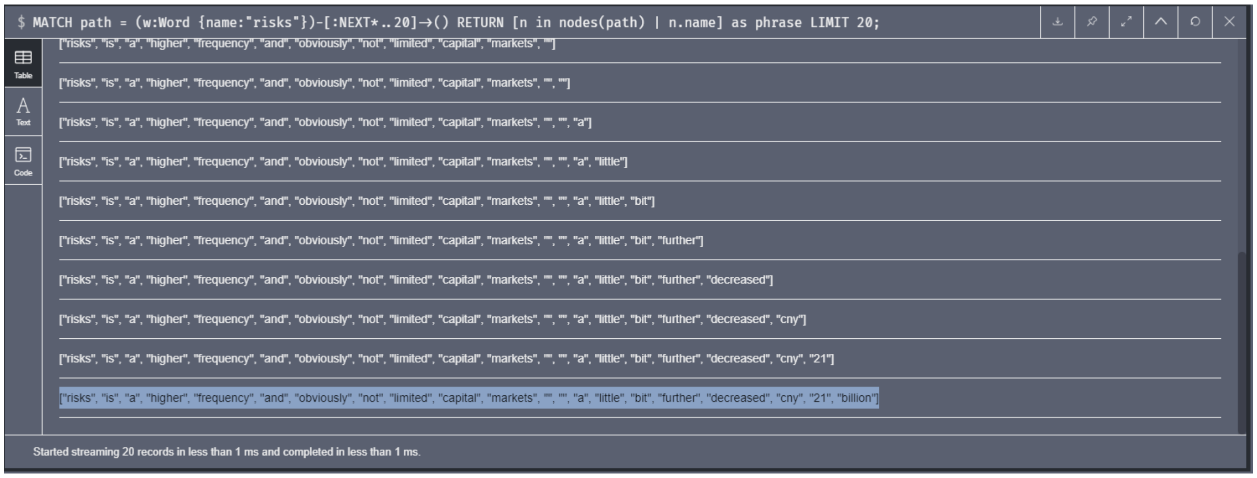
Cypher coding language in Neo4j was used to transform the ingested data into a graph database with each node representing a word from the Luckin Coffee earnings statement and the relationships between the words mapped out. Below is some of the code that was used to achieve this:

```
load csv from "file:///2019-Nov-13-LK.OQ-139494797335-Transcript.txt" as row fieldterminator ","
with row
unwind row as text
with reduce(t=tolower(text), delim in [",",".","!","?",'"',":",";","'","-"] | replace(t,delim,"")) as normalized
with [w in split(normalized," ") | trim(w)] as words
with [w in words WHERE NOT w IN ["the","an","on","to"]] as words
unwind range(0,size(words)-2) as idx
MERGE (w1:Word {name:words[idx]})
MERGE (w2:Word {name:words[idx+1]})
MERGE (w1)-[r:NEXT]->(w2)
```

Final Project Output

As part of our midterm submission, we used a python program to create a dictionary of n-gram (set to n=5 for this demonstration) strings and prompts the user to enter text based on these n-grams. Once submitted, the code automatically completed the remainder of the paragraph based on the text from the earnings statement.

For the final assignment, we were able to upload Luckin Coffee' earning statement into Neo4j in the form of a text file. From there, we used Cypher to predict a sentence sample of which has been included below:



```
$ MATCH path = (w:Word {name:"risks"})-[:NEXT*..20]→() RETURN [n in nodes(path) | n.name] as phrase LIMIT 20;

["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", ""]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", ""]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit", "further"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit", "further", "decreased"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit", "further", "decreased", "cny"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit", "further", "decreased", "cny", "21"]
["risks", "is", "a", "higher", "frequency", "and", "obviously", "not", "limited", "capital", "markets", "", "", "a", "little", "bit", "further", "decreased", "cny", "21", "billion"]

Started streaming 20 records in less than 1 ms and completed in less than 1 ms.
```

The reconstructed sentence shows each unique word used in the statement as a node as well as their ordinal relationship relative to a preceding or succeeding word from the statement.

The assignment detail document provides the step to step instruction on this project. The assignment detail document along with this document has been uploaded to our respective github sites the links to which can be found in the canvas submission.

Possible Next Steps

Potential next steps or "Part 2" enhancements to this that we considered:

- Increasing data set to include multiple earnings statements
- Including sentiment analysis of these statements and seeing how this correlates with stock price movements.
- Incorporating predictive text generation based on the larger dataset. This could be used in real-time as earnings are announced to make faster decisions on how to trade the stock.