Summer Placement Report
Microsoft Research Labs Asia

Ethan Ooi

9938120

MEng Computer Science (Hons)

One Week Automatic Extension Granted by DASS as part of my Study Support Plan

Introduction

This document will outline in detail the different tasks and responsibilities I overtook during the duration of my internship at *Microsoft Mobile Alliance Internet Services*, in the *Microsoft Research Lab Asia, Beijing, China*. I will describe in detail the technologies I have come in contact with, and contributed towards the company. There have been sections where potential solutions have been generalized, for the purpose of maintaining non-disclosure, and ensuring that no party takes a solution to market before the company does.

Microsoft Mobile Alliance Internet Services (MMAIS) was once a large department of Nokia stationed in Taiwan, and focussed on Maps and Location based services. In 2014, Microsoft Research Labs Asia bought the company, and transferred all its' employees into a new department, MMAIS.

Currently, the semi-autonomous department handles large projects of all types, with a large focus however, still being put on Maps, and in particular, management of Bing maps for the whole of mainland China and its' territories. Examples of other products that MMAIS have developed recently would include: Microsoft Office live integration and embedding of Bing Maps, A Graph-Database music recommendation Machine Learning Model, and close cooperation with Tesla, in providing the on-board navigation for the Chinese variants of their autonomous vehicles.

My internship began on the 24th of June 2019, and Concluded on the 23rd of August of that same year. The purpose of going into this internship was to gain hands on experience in working in the industry, in a field that is relevant to my professional development. This would involve me being a part of a team, working to deadlines, and applying my creativity to try and solve problems that were relevant to the project at hand. One other important skill that I hoped to get experience in were aspects of business development, analytical skills, team dynamics, and other entrepreneurial skills; Not directly taught in my course, but would be useful throughout my career working in the software industry.

Throughout the duration of my internship, a personal goal of mine was to try and add value to the team, and help in any way I could in furthering the development of the team as a whole. This would require me to identify specific strengths of mine that could synergize with the strengths of the team, to further empower the group.

One roadblock that was apparent as soon as I arrived was the language barrier. I took this as a challenge to myself, and set out to attempt all my communication strictly in Mandarin, with the hope that being fully immersed in this environment would improve my Mandarin speaking fluency and working proficiency.

Settling in

During my first week, it was apparent that one of the key skills that I could bring to the team was my ability in the English language. This was a key competence that brought value to the team, especially in the field of research. As I joined right at the beginning of a yearly project cycle, a lot of the work done at the time was revolving around research and feasibility studies for new clients and projects. I got placed loosely in two different but related projects: **Empowering a bank with Graph Knowledge Bases and Machine Learning,** and **Unstructured Natural Language Question and Answering of a Graph Knowledge Base.** Due to the state-of-the-art requirements of the products delivered, a lot of the newer research was only published in English at the moment. This is where my traits as someone who is proficient at english, and familiar and interested in the topics behind the technologies used in these topics, helped me become a useful member of the team.

And thus, a big part of my job was to set out and do research on the state-of-the-art, condense the information, and carry out critical thinking on if/how these new technologies could be applied to the problems at hand.
My two main sources of supervision and guidance would come from the CTO Ye Fei, and the department's AI and Machine Learning expert- Zhang Jing. My condensed research and ideas would go straight to them, which was good, as they were working directly with the clients involved with this project. My condensed research would materialize in the form of several reports and presentations, as well as notes on research papers that I would summarize for the Zhang Jing. Throughout this Placement Report, I will be summarizing the contents of each of the projects, as well as my contributions, in moderate technical detail.

The Bank Project

The first of the two projects I was involved in was for our client, China Construction Bank. As I joined for my internship, the team was in the process of determining how we could use Knowledge Graph technology to aid the bank in various of its' problems.

The second of which was a Graph-Knowledge-Base retrieval problem, trying to find a way to query this graph-database through unstructured natural language queries. This would involve using knowledge from learning about graph databases, as well as NLP to find a way to bridge the gap between user-intent and query-construction, and allow the creation of a generalized Question/Answer assistant.

Most of my focus during my short two month placement was placed on the first project, as it was given by an external client, and it was of utmost urgency that research and investigation into the available technologies be thoroughly completed before any further development can occur.

The Banks' Problems

After several meetings between my department and the China Construction Bank (CCB), several key problems were brought up, of which we went on to investigate the capability of us providing viable solutions.

The list of problems that the bank believed we could assist them in is as follows:

a) Financial risk management for both existing customers and enterprises which they had not dealt with before through user profiling or pattern recognition;

b) Financial credit check for Micros companies (小微企业) nationwide for loan services;

c) Risk assessment for house loan service;

d) HR application to help their staff to sell CCB products to customers. I was guessing that this may be a joint effort involving Dynamics for sales with AI capabilities.

e) Digital asset management;

CCB were interested in trying to levy graph technology to be able to perform better analytics on its' datasets. The bank would want my department to present an appropriate technology, along with the ontological design of its' data required to fully utilize the solution.

The size of their knowledge graph would contain about 1 Billion Entities, with 500 million relationships. We had to find a scalable and accurate way to carry out graph-embedding, in order to provide a scalable solution to handle the fast-growing middle-class in China.

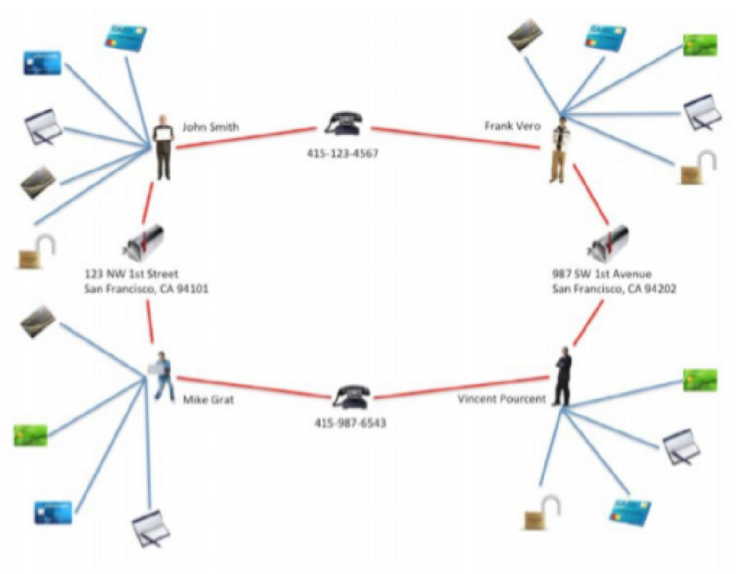
The main problem highlighted by CCB, is that we would be working with a lot of missing or unknown values. This is due to a regulatory privacy requirement, that prevents even national banks to receive the financial history of new clients, from their previous banks. This complication would open up the possibility of performing link-prediction tasks, or correlating fiscal responsibility through temporal or relational data.

The ultimate goal for CCB would be to minimize losses from bad loans and fraudsters. Whilst accomplishing this, graph technology can allow for lineage tracing of decisions, which is a new regulatory requirement presented by international finance institutions for accountability.

Fraud Rings

I began searching through different methods that the bank can increase its value by reducing the number of defaulted loans. I came across a wealth of research on how graph databases can be used to detect deep fraud rings.

In the US, first party bank frauds account for over 25% of all bad debt. This is a significant of money that could be saved by detecting fraud before these fraud rings strike out. A ring of fraudsters create synthetic identities by sharing and combining various personal attributes, in order to create accounts that will slowly build up their credit lines, before suddenly ‘breaking-out’, and maxing out all of their respective credit lines.



Here we can see how just using 2 addresses, and 2 phone numbers, fraudsters can create 4 different synthetic identities.

This is a very common method of maximizing the money extracted from a bank through fraud.

As the size of these fraud rings increases, the number of synthetic identities, and thus the amount lost by the banks, rise exponentially.

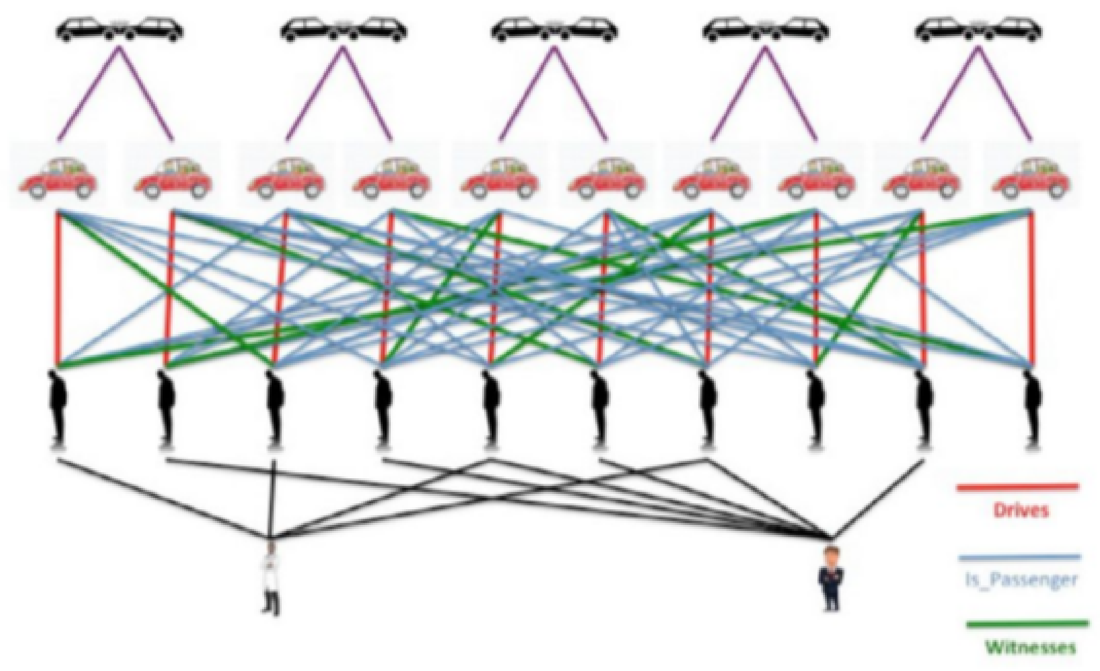
With just three fraudsters, the number of synthetic identities rises to 9. Catching these types of connections is far harder with traditional relational databases, with the complexity of all the Inner and Outer Joins to create a huge amount of tables; growing exponentially in computational complexity with the increase in fraud-ring members, and the combinations of shared information they utilize.

However, through utilizing graph databases, simple walks and graph traversals are able to identify a large number of these (literal) rings.

Another advantage is that it is easier for staff to manually review these accounts, as fraudulent ring shapes are more apparently visible.

A similar fraud is carried out with insurance fraud, with people participating in 'paper-crashes' with fake victims, witnesses, and mechanics misreporting actual costs. Fake doctors are also used to mis-report the actual damage and treatment costs for these fake victims.

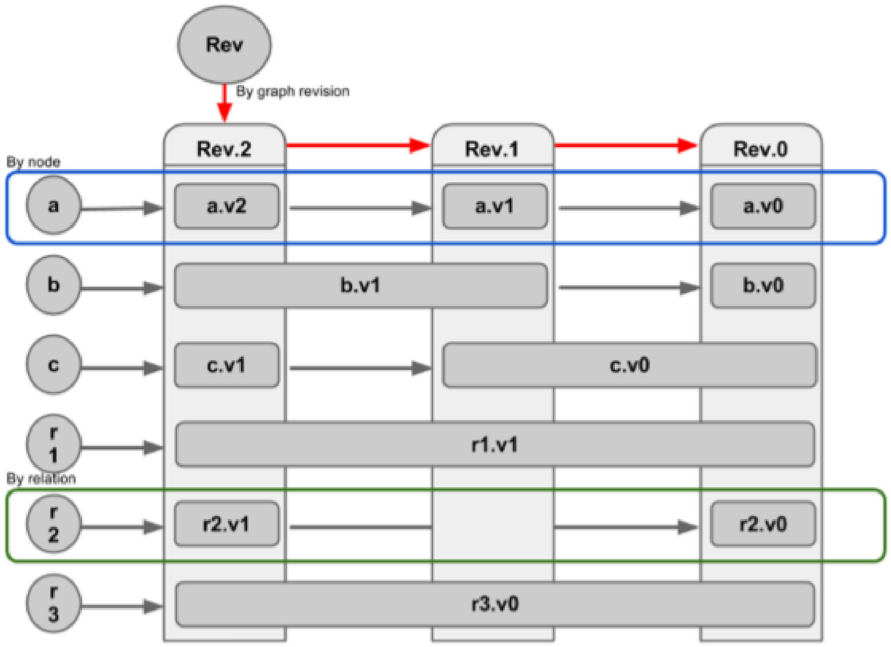
The impact of fraud on the insurance industry is estimated to be $80 billion annually in the US, a number that has been growing in recent years. From 2010 to 2012, questionable claims in the U.S. jumped 27 percent, to 116,171 claims in 2012, nearly half resulting from faked or exaggerated injury claims.



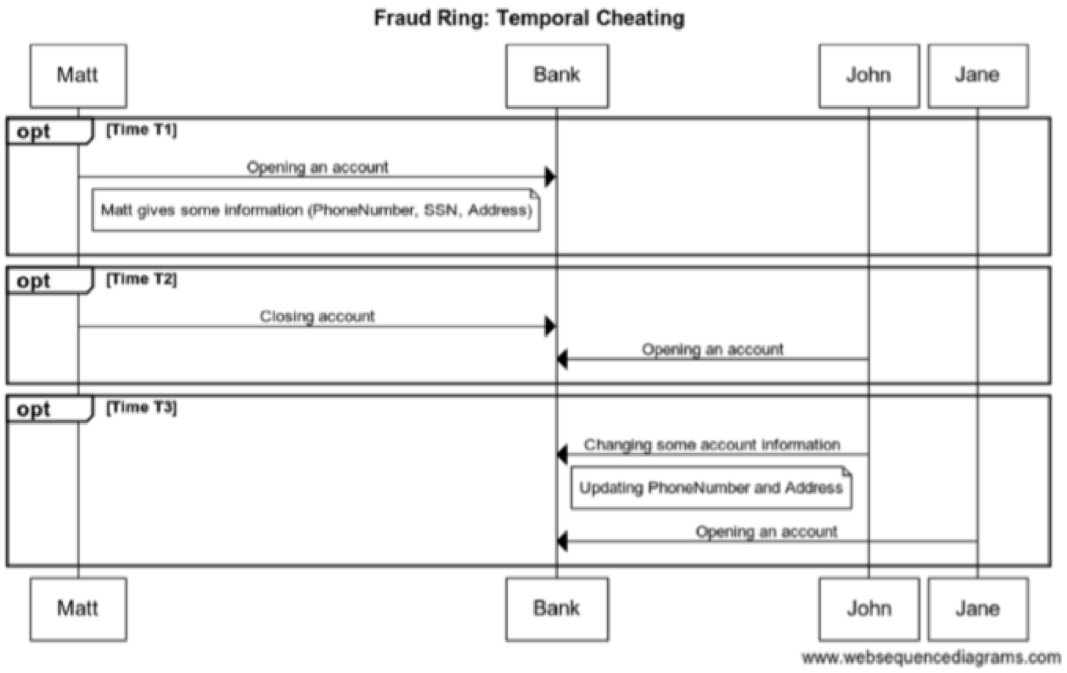
Here, we can see that 10 people can collude to carry out 5 'paper-crashes', "In an example where ten people collude to commit insurance fraud, five false accidents are staged, where each person plays the role of the driver once, a witness once and a passenger three times. Assuming an average claim of $40K per injured person and $5K per car, the ring can claim up to $1.6M for 40 people injured!"

Using graph databases, it is far easier to see these frauds, which emerge not as a consequence of datapoints, but as a result of relationships. Graph Databases are the intuitive way to represent data in an industry where relationships between entities contain the most significant insights. Complex n-order relationships in which fraudulent users disguise themselves can be detected with far less computational complexity than traditional methods, possibly leading to the apprehension and persecution of more sophisticated fraudsters using relationships that are much harder to dis-entangle using traditional relational database techniques. [5][6][7]

Another way that smart fraudsters can be caught, is through temporal version analysis of a database. In the paper **'Rogue behavior detection in NoSQL graph databases' [4]**, we can observe a version graph view of the database, to detect suspicious behavior that would otherwise be invisible if viewed simply at any current slice in time.



History is a key feature in discovering fraud rings, and fraudsters are beginning to play with delays in order to gradually root their actions. However, time is not always represented in the systems used to store information, which prevents organizations from effectively wiping out fraud and corruption.


Fraud Ring: Temporal Cheating
Matt
Bank
John
Jane
opt [Time T1]
Opening an account
Matt gives some information (PhoneNumber, SSN, Address)
opt [Time T2]
Closing account
Opening an account
opt [Time T3]
Changing some account information
Updating PhoneNumber and Address
Opening an account
www.websequencediagrams.com

A temporal view of the dataset can not only aid manual review, but also be used in conjunction with cycle detection algorithms to increase the analysis of the data-set by one dimension.

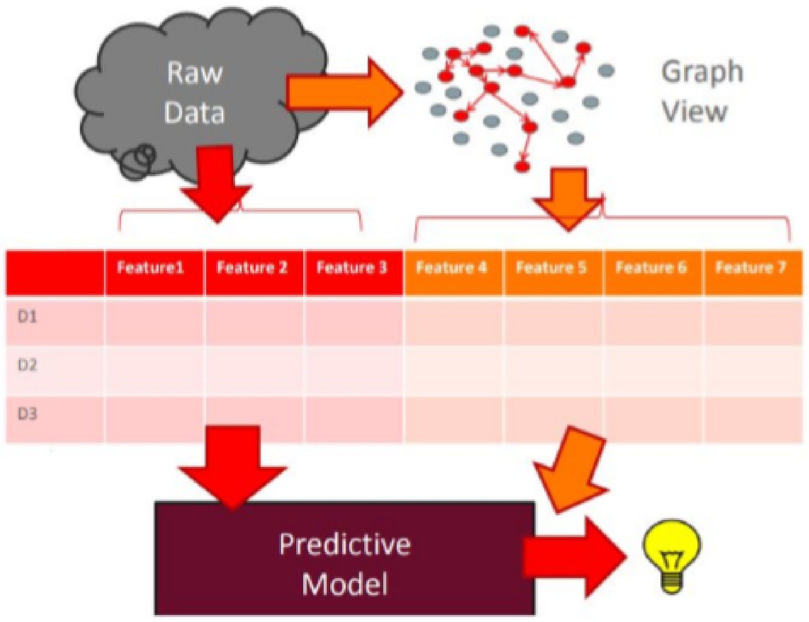
Financial Credit Estimation

Current financial analytic techniques currently have a large unsolved problem in estimating how willing and able businesses shareholders and decision makers are to repay their loans in time. This problem is exacerbated by the fact that in China the required information is difficult to acquire, and often takes upwards of more than one month, which is far too long for banks to receive this crucial data, and make a timely decision.

A bank in China will have great difficulty in gathering information about the client's history with other banks, so new customers coming to the bank will have to all go through manual review with very limited information, which predictably leads to poor results in actually determining the credit-worthyness, or business success likelihood, of a new prospective clients.

" Humans are not good at making credit decisions. A large part is based on the first few seconds of how much they like the person." -Forbes 2011

Traditional predictive models, use metrics such as Profit/Loss, Debt/Equity, FICO score, and so on as inputs to their predictive model. However, through my research, it can be shown that we can combine the power of Graph Models, as well as current Predictive Models, to increase the sensitivity of our prediction. This can be done through using a graph view of the data, to synthesize additional relationship-based features to use as input to the predictive model. [9]



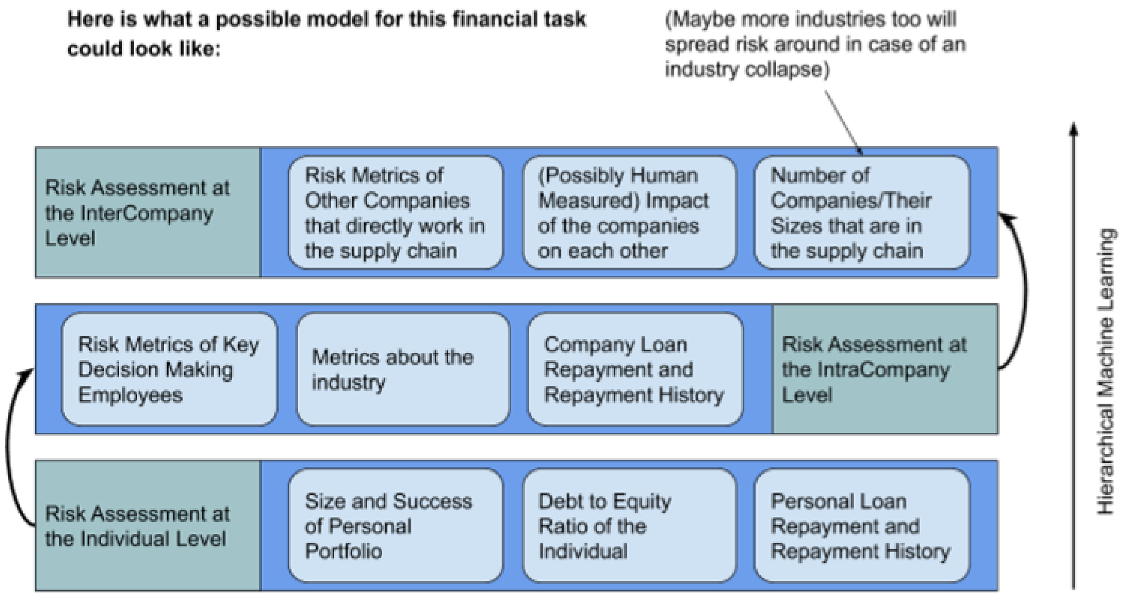
However, we are still far from meeting new data, privacy, and financial regulations imposed by global financial institutions. With an increase in financial regulations such as **BCBS239** and **MiFD II**, data lineage is required more than ever, as heuristics based ad-hoc in-house risk reporting methods are exposing the limitations of existing compliance systems to conform to the new regulations. [8]

With the recent addition of GDPR, which largely states that: Data must be governed at an enterprise level, across every line of business, in every jurisdiction where business is conducted. Through these regulatory changes, we see that it is important for banks to have a

more transparent system through which they process data, and create a decision. This is one aspect that a graph view could help banks abide by these regulations, by tracing decision lineage.
I proposed a model, inspired by the risk lineage tracing of trading desks inside the financial market, first proposed in a white-paper written by the Neo4j team, titled "**Addressing Key Challenges in Financial Services with Neo4j**" [1].

A key concept taken away from this for me was in a method for minimizing stock trading risks at the trading desks, which involved first tracing the risk data lineage;
from **Trading Desk → Baskets → Industry → Stock Prices**. After which we can aggregate the total risk back down in the other direction, to get a more Cause/Effect view of exactly how risk traces through the system.

With this in mind, I proposed a model where we can predict the overall risk of new emerging businesses, by aggregating risks in the following manner:
Key Decision Makers Risks → Company Aggregate Risks → Risks of the Companies they interact with


Here is what a possible model for this financial task could look like:
(Maybe more industries too will spread risk around in case of an industry collapse)
Risk Assessment at the InterCompany Level
Risk Metrics of Other Companies that directly work in the supply chain
(Possibly Human Measured) Impact of the companies on each other
Number of Companies/Their Sizes that are in the supply chain
Risk Metrics of Key Decision Making Employees
Metrics about the industry
Company Loan Repayment and Repayment History
Risk Assessment at the IntraCompany Level
Risk Assessment at the Individual Level
Size and Success of Personal Portfolio
Debt to Equity Ratio of the Individual
Personal Loan Repayment and Repayment History
Hierarchical Machine Learning

This model borrows from an application in chemistry of Hierarchical Machine Learning to predict the properties of a material, based on **Properties of the atoms → Properties of the structure→ Complex System responses- i.e Stress/Strain/Shear Strengths [10]**
Through my proposed model, it allows for the possibility to trace the lineage of a financial decision, and identify exactly how risks flow upwards through the different factors affecting the loan decision. Although this does not solve the problem of a lack of powerful predictive features,

it may be an idea to consider to greater empower what little features are available, in making loan decisions for micro-companies with no prior credit history with the bank.

Feature Selection

Another subject of my research for Financial Graph Databases was on investigating commonly used features in predicting loan repayment.
One of the most common methods in feature selection is using a decision tree to weigh the effects of each attribute on the overall predictive outcome. In a small dataset of only n=9500, one researcher used a Random Forest Classifier to determine the importance of each feature.

The results [3] are as follows:
The data covers the 9,578 loans funded by the platform between May 2007 and February 2010
1. The number of days the borrower has had a credit line.
2. The interest rate of ethe loan (proportion).
3. The monthly installments ($) owed by the borrower if the loan is funded.
4. The debt-to-income ratio of the borrower.
5. The natural log of the annual income of the borrower.
6. The borrower’s revolving balance.
7. The borrower’s revolving line utilization rate.
8. The FICO credit score of the borrower.
9. The borrower’s number of inquiries by creditors in the last 6 months.
10. Purpose of loan- Debt consolidation
11. The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
12. Purpose of loan- Credit Card
13. Purpose of loan- Small Business
14. The borrower’s number of derogatory public records.
15. Purpose of loan- Home Improvment
16. Purpose of loan- Major Purpose
17. Purpose of loan- Educational

While these seem to be fairly standard metrics required to predict loan repayment internationally, it is quite a bit more complex in China, as a lot of this information is either inaccessible, unavailable, or non-existent.

To try and find a method that might work for China, I looked towards Kaggle’s $70,000 Prediction Competition on a loan repayment dataset (n=357,000). One of the top scorers[13] used a method dubbed ‘Deep Feature Synthesis’ [2], which combined features in various ways in order to collate them into groups that had the most predictive power. The results were an impressive ~81% predictive accuracy of loan repayment on the testing set (n=39,000). The reason this might work for China, is because Deep Feature Synthesis will take all the features you are able to give it, and create new compound synthetic features for use in predicting the outcome.

Graph Embedding

Following my research on the high level methods that can be employed, I began to look at different state-of-the-art graph based machine learning techniques from the year 2018, with the goal of investigating the feasibility of suitable models for use in our use case.
Graph Embedding was the main topic of my research, which is the process of turning a graph view of the data, into a number based summary of various aspects of the graph, to be used for computerised reasoning, or decision making.
I focussed my research on the following papers:

ConvE- Convolutional 2D Knowledge Graph Embeddings

(July 2018)[14]
The paper presented a ML model, which they dubbed **ConvE**. The goal of ConvE is to predict missing relationships between entities. The key idea is, scoring all the possible missing links, and finding which ones are likely to be true, but missing from the Ground Truth dataset. A knowledge graph **G** is a collection of triples **(s,r,o)**, where **s and o** are entities, and **r** is a relationship. The traditional brute-force approach would be to take the set of **(∀s, ∀r, ∀o)**, and individually score each of them, which is a **1 to 1** mapping of the triples to their respective scores. However, The research team's technique is to take **∀(s,r)** ∈ **G**, and score them against all entities **o**, which is instead a **1 to N** mapping, and can be parallelized. The first method took the researchers **3.35h** to execute their link prediction, whilst the second method only took **35 seconds**. The authors of the paper claim that ConvE is good for modelling nodes that have a high recursive in-degree.

KG2Vec - Expeditious Generation of Knowledge Graph Embeddings

(November 2018)[15]
KG2Vec is a Knowledge Graph embedder based on the Natural Language Processing model- the SkipGram model. Its key advantage over other embedders is the ability to embed large graphs into vector-space, efficiently and quickly. It uses an LSTM network to score embeddings by using a neighbor similarity distance metric in n-dimensional vector-space.

The results this group of researchers have received have not been impressive in terms of accuracy, and appears to perform poorly at the task of link prediction. The researchers claim that their model may benefit from additional training, and that the aim of their study was to generate embeddings at a high rate, while preserving some semblance of accuracy.

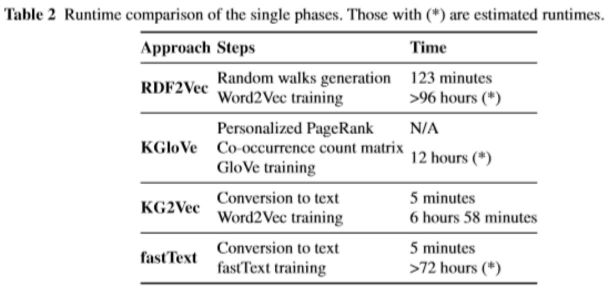


Table 2 Runtime comparison of the single phases. Those with (*) are estimated runtimes.

Approach	Steps	Time
RDF2Vec	Random walks generation Word2Vec training	123 minutes >96 hours (*)
KGloVe	Personalized PageRank Co-occurrence count matrix GloVe training	N/A 12 hours (*)
KG2Vec	Conversion to text Word2Vec training	5 minutes 6 hours 58 minutes
fastText	Conversion to text fastText training	5 minutes >72 hours (*)

Their model does appear to perform quickly at both training, and generating embeddings, and may be promising in use-cases where speed is prioritized over accuracy.

Deep Path

(July 2018) [16]
Deep Path is a model based on Reinforcement Learning, for learning relational paths in Knowledge Graphs. It utilizes a compound reward function, which optimizes for accuracy, efficiency, and path diversity. The specific task of this method is to find reliable, predictive paths between entity pairs. We formulate the path finding problem as a sequential decision making problem, which can be solved with a Reinforcement Learning agent. The RL agent learns to pick promising reasoning paths by interacting with the KG environment. This interaction is done through picking actions (entity pairs **(e_s,e_t)**), which changes the state of the environment, which is modelled after a Markov Decision Process, as the tuple **<S,A,P,R>**, where **S** is the continuous state space, **A** is the set of all available actions, **P** is the transition probability matrix, and **R(s,a)** is the reward function for every **(s,a)** entity pair.
The RL agent is represented as a policy network, $\pi_\theta(s,a) = P(a \mid S;\theta)$, which maps the state vector to a stochastic policy. The NN parameters θ are updated using stochastic gradient descent.
The rewards are summarised as follows:
Global Accuracy: Offline reward of +1 given to the agent that reaches the target after a sequence of actions, -1 otherwise.

Path Efficiency: For the relation reasoning task, the researchers observe that short paths tend to provide more reliable reasoning evidence than longer paths (Better Lineage Tracing), hence the reward is inversely proportional to the length of the sequence of actions taken by the RL agent.

Path Diversity: The agent tends to find paths with similar syntax and semantics, which often contain redundant information, as some of the paths may be correlated. To encourage the agent to find diverse paths, the researchers define a diversity reward function which is inversely proportional to the cosine similarity measure between the current path, and existing paths. This encourages the agent to try paths that are not already present in the ground truth.

Training is carried out using a randomized Breadth First Search (BFS), in order to reduce the size of the set of possible actions, which can be in the order of thousands of possible initial actions for the agent.

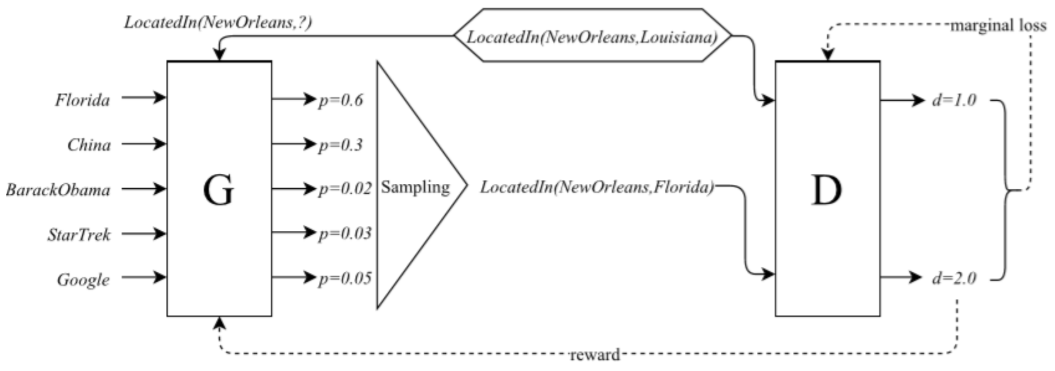
KBGAN- Adversarial Learning for Knowledge Graph Embeddings

(April 2018) [17]
One big problem on training models is, although we have a set of ground truth paths that are already present inside the Knowledge Graph, it is much harder for us to produce Negative training samples. To illustrate this, we can imagine the example of trying to create a fake triple to

train any model. Merely matching Entities with relationships randomly will yield triples that are too easy to distinguish from reality, i.e **“Barack Obama, Ingredient Of, Orange Juice”**.
This is the current method of generating negative triples for training, just selecting entities, or relationships at random.

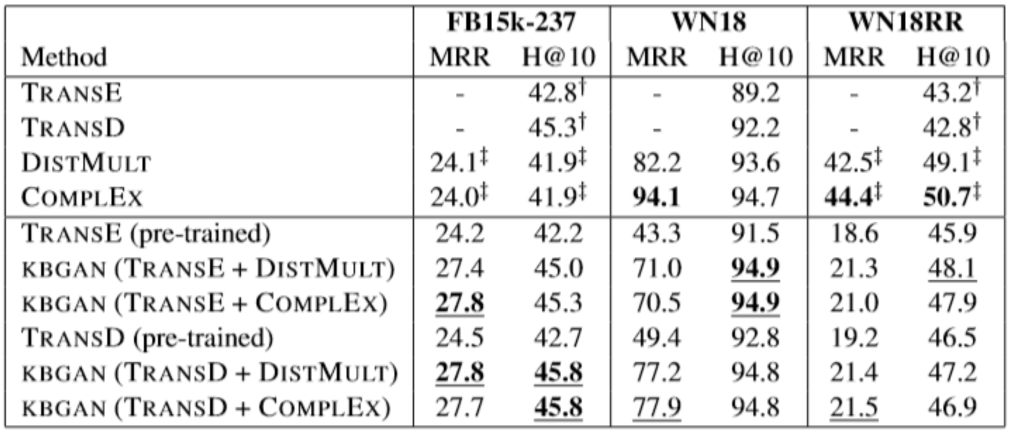
This paper discusses a method of using a Generative Adversarial Network (GAN) to train a generator to create believable negative triples, i.e **“Steve Austin, Lives In, Michigan”.**

The problem with traditional GANs for this approach is that the discrete sampling step- the Discriminator reducing the continuous probabilities of a decision, prevents regular gradient descent from propagating change back into the generator.
The solution proposed is to use a reward function, based on some similarity measure, to flow back into the generator for training. The Discriminator should assign a relatively small distance to a high-quality negative sample.



As seen by the above diagram, the Generator selects candidate entities for use in creating a fake triple. The discriminator receives both the real triple, and the fake generated triple, and assigns a distance metric on how similar the triples appear to be. The goal of this is to have the generator create triples that have high concordance to the properties of the ground truth dataset, with the final goal of being able to create negative triples that are indistinguishable, to the discriminator, from the ground truth.
The result is, a Discriminator and Generator that are both well trained at detecting bad fake samples, and a Generator that is good at generating good Fake samples, with high correlation to the original data set. We then use this well trained Generator in order to create negative training samples, for use of any of our graph embedding models.

The research team tested the generic solution on a wide range of embedding models, and it seemed to improve the quality of the predictions made by those models by about 5%, which boosts it to state-of-the-art status in many cases.



Method	FB15k-237 MRR	FB15k-237 H@10	WN18 MRR	WN18 H@10	WN18RR MRR	WN18RR H@10
TRANSE	-	42.8†	-	89.2	-	43.2†
TRANSD	-	45.3†	-	92.2	-	42.8†
DISTMULT	24.1‡	41.9‡	82.2	93.6	42.5‡	49.1‡
COMPLEX	24.0‡	41.9‡	**94.1**	94.7	**44.4**‡	**50.7**‡
TRANSE (pre-trained)	24.2	42.2	43.3	91.5	18.6	45.9
KBGAN (TRANSE + DISTMULT)	27.4	45.0	71.0	**94.9**	21.3	48.1
KBGAN (TRANSE + COMPLEX)	**27.8**	45.3	70.5	**94.9**	21.0	47.9
TRANSD (pre-trained)	24.5	42.7	49.4	92.8	19.2	46.5
KBGAN (TRANSD + DISTMULT)	**27.8**	**45.8**	77.2	94.8	21.4	47.2
KBGAN (TRANSD + COMPLEX)	27.7	**45.8**	77.9	94.8	21.5	46.9

Overall Takeaway

My overall impressions on these 4 techniques is that **KBGAN** can be a very powerful tool, when used in conjunction with other Graph Embedding models. The conclusion I presented was that it may be worth investigating a combination of ConvE, with empowered training from KBGAN. Because of ConvE's ability to perform well at modelling nodes with high in-degrees, I believe it is well suited for the application of modelling risk lineage and deep-fraud-rings. ConvE is able to model more complex relations, which can be useful in detecting more sophisticated fraud rings that would otherwise be invisible to manual review.

Deep Path with its' ability to find diverse paths, may be able to link entities that would not have otherwise thought to be linked, and due to its' RL based methodology, the sequence of actions is transparent, apparent, and easily visible. This will aid in lineage and transparency when it comes to link-prediction as a part of completing an incomplete graph for Fraud Detection.

My findings were presented to the team in moderate technical detail, and an implemented solution is still in the process of being worked out, as the job is expected to be completed over this new fiscal year. My supervisor and CTO was especially interested in KBGAN's ability to empower the training of other models.

Programming to support learning

The papers I was asked to research were currently far above my ability to implement these Machine Learning techniques to contribute to the codebase, however I have used this as an opportunity to learn about the concepts behind these state-of-the-art graph embedding techniques.
In between researching papers, I would attempt to build related ML Architectures I was unfamiliar with, in order to better understand the limitations of the solutions that could be presented.
I used Python with Keras and Tensorflow to implement my various models.

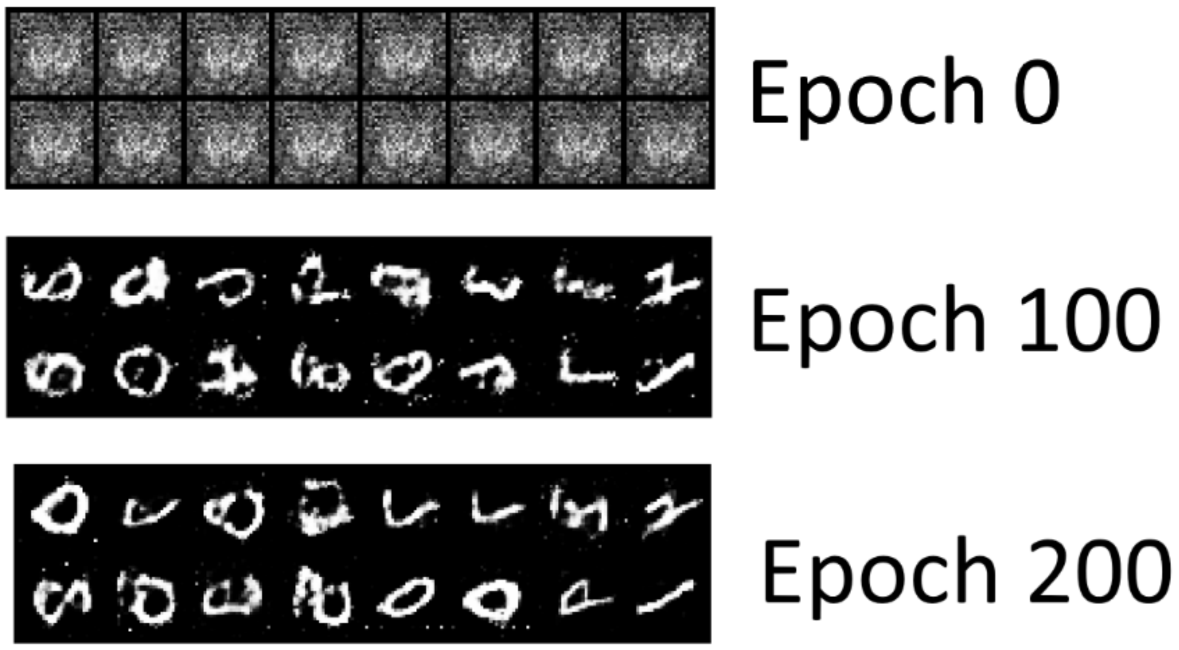
MNIST
I first began with experimenting with various ways to solve the MNIST Digits dataset. I would implement Convolutional Networks, Deep Neural Networks, and Random Forest Classifiers. I tinkered with the construction of the layers to compare both training speed, loss, and testing/validation accuracy.

I learned that deep analysis had to be done on why each of the layers was chosen which would depend on the nature of the data itself. I learned first-hand how much tinkering was required to increase the accuracy of the model. Even in a basic sequential network, there are so many hyper-parameters that need tuning, these include: Number of Layers, Number of Neurons in each Layer, the Activation Function for each layer, The selected loss-function to evaluate progress, and the optimizer type to tune weights.

Through my tinkering, I learned more about the first-hand complications of fine-tuning neural networks to fit specific problems. A key takeaway I got from this task was that network design is an extremely complex task, and simply throwing more layers onto the network would not make the model any more performant.

GAN
Generative Adversarial Networks were the topic of one of the research papers I studied. In order to get a better grasp on their limitations, I implemented my first GAN on the MNIST dataset. The goal of my network would be to train a Generator to generate realistic handwritten digits from 0-9, and to train a Discriminator to evaluate the quality of these samples.
Getting the GAN to work required an immense amount of tinkering. This was the first time that I had combined the use of two different Neural Networks, the Discriminator, and the Generator. For each of these networks, I would have to tweak the architecture of each of their layers, in order to optimize for training speed (loss convergence) and accuracy (loss value).

The GAN takes as input a normal-distribution of random noise, and the trained Generator would modify it into an image of a digit ranging from 0-9. My model converged on a relatively high loss value after a staggering 200 epochs (Took half a day to run). This indicates that my model needed far more tuning in order to perform well.



Text Analysis
My next personal challenge was to familiarize myself with the ML techniques used to deal with text. The dataset I chose to learn with was the IMDB reviews dataset. The task at hand was to determine if a given review had a negative or positive sentiment. This involved me learning about Text-Embedding, implementing Word2Vec, and then experimenting with using this vector as input to a Convolutional Neural Network. I also used the outputs of Word2Vec to implement a Long-Short-Term-Memory (LSTM) Network, and a Recurrent Neural Network.
These two networks are theorized to work especially well on text, as they both have some concept of a sample-history, or short-term-memory, which would allow the network to “remember” words and context in a body of text. I was able to perform sentiment analysis using these three network architectures. As expected, the LSTM and RNN outperformed the CNN greatly. The ability to remember the context of words better equipped these networks to parse an entire body of text.

Working with text was new and foreign to me, and required me to gain proficiency in working with un-cleaned data, and further processing (vectorizing) the text in order to be able to use them as inputs to my models. Learning about vectorization has taught me a whole new way of thinking about text-data, and has given me more insight on how to turn our messy complicated world into a standardized data-format for use in our precise and obtuse computers.

Numer.AI
My Supervisor Ye-Fei had suggested that I start honing my skills on various Machine Learning competitions. He suggested that I try entering a bi-weekly finance ML competition- numer.ai This competition provides clean data-sets every round, and has users wager money on a stock-market prediction problem.

This was quite a challenge, and I first tried approaching it with standard Feed-Forward Neural Networks, and Convolutional Networks. However, this produced results on the tournament dataset that were highly unsatisfactory. After tinkering with my models for a period of time, I conceded and went to the ML expert of our department for guidance. He suggested that I look at Gradient Boosting as a means to enhance simpler models. This was because the over 200 features appeared to closely co-correlate, and a traditional Neural Network would have to be extremely large and deep in-order to capture the feature behavior.

Gradient Boosting would allow for my network to converge on a loss value sooner, and hopefully allow me to make better predictions with respect to training-time-constraints of my personal laptop. I Implemented a Gradient Boosted Decision Tree to tackle the dataset. This allowed me to pass the correlation cutoff required to even enter the tournament, in which I placed fairly poorly: 203th out of 460 entries, however this was a start, and gave me more practice in classifying a large dataset.

What I took away from this was that I was still very far away from being able to bring my model-creation to the competitional level, however learning of numer.AI has given me a place to benchmark my skills against other ML engineers. I had a lot of fun tinkering with my initial Neural Networks, and have realized that as I gain more experience, I will be better equipped to select the right Architecture for the right problems, given the resource constraints.

Last known state of project and my takeaway
As my placement was only 2 months in duration, I was not able to witness the completion of this project, as it is still currently in progress. When large sums of money are involved, the project life cycle tends to be longer, as tasks such as feasibility tests, integrating the technical team of the bank, and extensive deliberation and product specification, are required to minimize the risk of a project of this size.

One huge takeaway for me from this project was getting more familiar with the financial sector. Before this, I had never fully realized the extent of financial losses from mismanaged risks. Working with Microsoft and China Construction Bank has given me a first-hand insight on how technology is evolving to keep up with the cat-and-mouse game that arises from intelligent fraudsters.

My accomplishments from participating in this project were digesting cutting-edge technology, and presenting my findings in Mandarin to my team. I have also been involved with meetings with the bank, and got to contribute to investigations and client deliberations, which has taught me about how business works with big clients in China, and the very particular way that business needs to be done with the Chinese Government.

I am extremely fortunate, being exposed to a wealth of cutting edge technology, as well as having the privilege to receive the help of field-experts. Through these experiences, I have pushed my theoretical understanding of more complex Machine Learning models. I have also gained a lot of first-hand guidance on the nuances of tuning my own models. This is the type of hands on guidance and experience that I was looking for, and the lessons learned here will definitely be an asset to any future projects throughout my career.

The experiences from this project alone has without a doubt increased my value as a software engineer. I believe the invaluable business experience gained is an especially strong asset, and has contributed greatly to my entrepreneurial skills, which is of incredible importance when it comes to making analytic decisions.

The Q/A Project

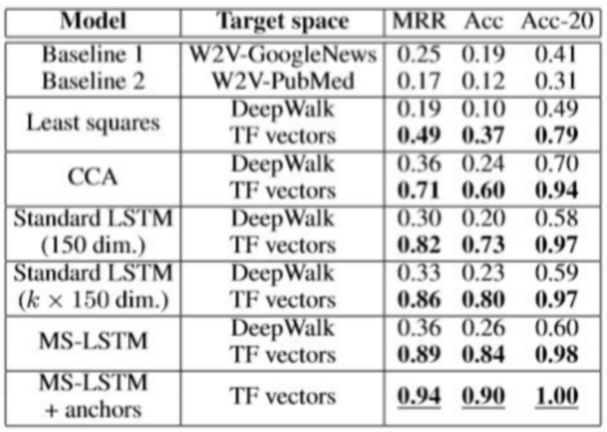
The other project that I had a bit of involvement in was the task of **Answering Unstructured Natural Language Questions by Querying a Knowledge Base**.

A simple example of this task, is to ask in natural language, the query “What movies has director Paul Anderson produced in before the year 2000”. It is a complex task that would first involve breaking down the semantic structure of the query, and then translating that into a graphQL query of the knowledge base. Key pitfalls could include disambiguation of entities, and distinguishing cases of Polysemy, and Homonymy.

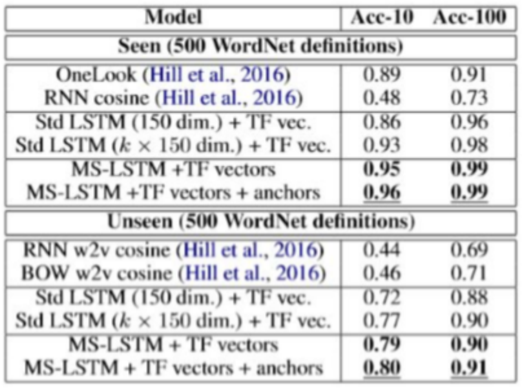
I began my research into this topic by looking into word disambiguation, and came across a paper on making a presentation on Multi-Sense LSTM networks to solve this problem.[11]

The basis of the MS-LSTM is to create a set of reliable anchors inside the KB space, which are one-to-one correspondences between the two representations of textual, and vector space.
We create these anchors by first extending the original KB graph. A simple method of doing so is by taking random walks along the KB graph, which are fed to a skipgram[12] model for producing an entity space. The traditional LSTM is extended with an attentional disambiguational mechanism that dynamically selects and updates the right sense vector for each word, given it’s context, during training.

The result is the creation of additional **textual nodes**, which are both words, and KB entries, which allows for an extremely effective job at transforming the geometry of the entity space to the benefit of mapping textual ambiguity. These textual nodes bring related concepts closer together through a path through the textual node, and allows for the KB to better group itself into domains of knowledge, making it especially effective at disambiguating between words based on context.
Points that would usually be unjustifiably far apart in vector space topology, are now brought closer together, providing additional coherence, and therefore stronger predictive ability through graph traversal using methods such as Deep Walk. Shorter walks generally indicate that the two entities in question are more related. The addition of textual features allow otherwise longer walks to be reduced in length, and rise to the top of relevancy measure.



Model	Target space	MRR	Acc	Acc-20
Baseline 1	W2V-GoogleNews	0.25	0.19	0.41
Baseline 2	W2V-PubMed	0.17	0.12	0.31
Least squares	DeepWalk	0.19	0.10	0.49
	TF vectors	**0.49**	**0.37**	**0.79**
CCA	DeepWalk	0.36	0.24	0.70
	TF vectors	**0.71**	**0.60**	**0.94**
Standard LSTM (150 dim.)	DeepWalk	0.30	0.20	0.58
	TF vectors	**0.82**	**0.73**	**0.97**
Standard LSTM ($k \times 150$ dim.)	DeepWalk	0.33	0.23	0.59
	TF vectors	**0.86**	**0.80**	**0.97**
MS-LSTM	DeepWalk	0.36	0.26	0.60
	TF vectors	**0.89**	**0.84**	**0.98**
MS-LSTM + anchors	TF vectors	**0.94**	**0.90**	**1.00**



Model	Acc-10	Acc-100
Seen (500 WordNet definitions)		
OneLook (Hill et al., 2016)	0.89	0.91
RNN cosine (Hill et al., 2016)	0.48	0.73
Std LSTM (150 dim.) + TF vec.	0.86	0.96
Std LSTM ($k \times 150$ dim.) + TF vec.	0.93	0.98
MS-LSTM +TF vectors	**0.95**	**0.99**
MS-LSTM +TF vectors + anchors	**0.96**	**0.99**
Unseen (500 WordNet definitions)		
RNN w2v cosine (Hill et al., 2016)	0.44	0.69
BOW w2v cosine (Hill et al., 2016)	0.46	0.71
Std LSTM (150 dim.) + TF vec.	0.72	0.88
Std LSTM ($k \times 150$ dim.) + TF vec.	0.77	0.90
MS-LSTM + TF vectors	**0.79**	**0.90**
MS-LSTM + TF vectors + anchors	**0.80**	**0.91**

As we can see, TF (Textual Features) provide a big increase in categorical accuracy

Subgraph Matching

I would next investigate the possibility of directly mapping a natural language query into a graph matching problem. The following two papers present a similar multi-step pipeline to directly match natural language queries to components of a Knowledge Base, in order to return results for a query.

Svega: Answering Natural Language Questions over Knowledge Base with Semantic Matching

[18]

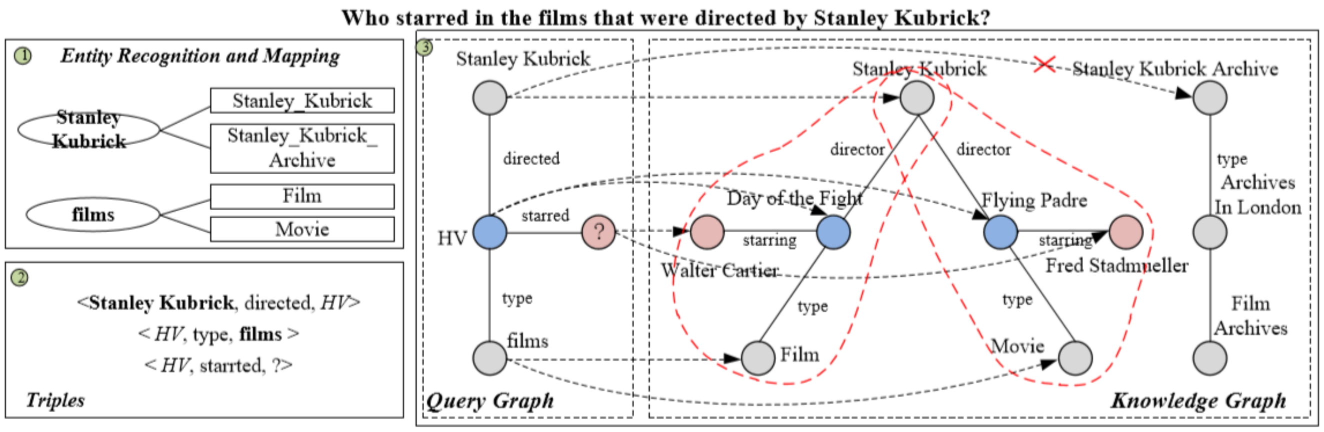


Fig. 1: System Overview

The first part of **Svega** constructs an Entity-Driven Graph in order to model the intention of the natural language question given. We extract triples from the query, and join the triples in order to construct the Query Graph. The next problem is to disambiguate candidates in the Knowledge Base, and match them to their respective Query Triples.
"The Hidden Vertices (HV) represent fuzzy entities, which allows the search to disambiguate by looking at the relationships contained by the ambiguous entities, and compare that to the candidate entities denoted by the Knowledge Graph."
Once matches have been found for each hidden vertex, disambiguation has been completed. The result to the requested query can then be easily determined.

TABLE I: QALD-3 on DBpedia 3.8

	Proceed	Right	Partially	Recall	Precision	F-measure
CASIA	52	29	8	0.36	0.35	0.36
Scalewelis	70	32	1	0.33	0.33	0.33
RTV	55	30	4	0.34	0.32	0.33
Intui2	99	28	4	0.32	0.32	0.32
SWIP	21	15	2	0.16	0.17	0.17
Svega	96	44	9	**0.52**	**0.52**	**0.52**

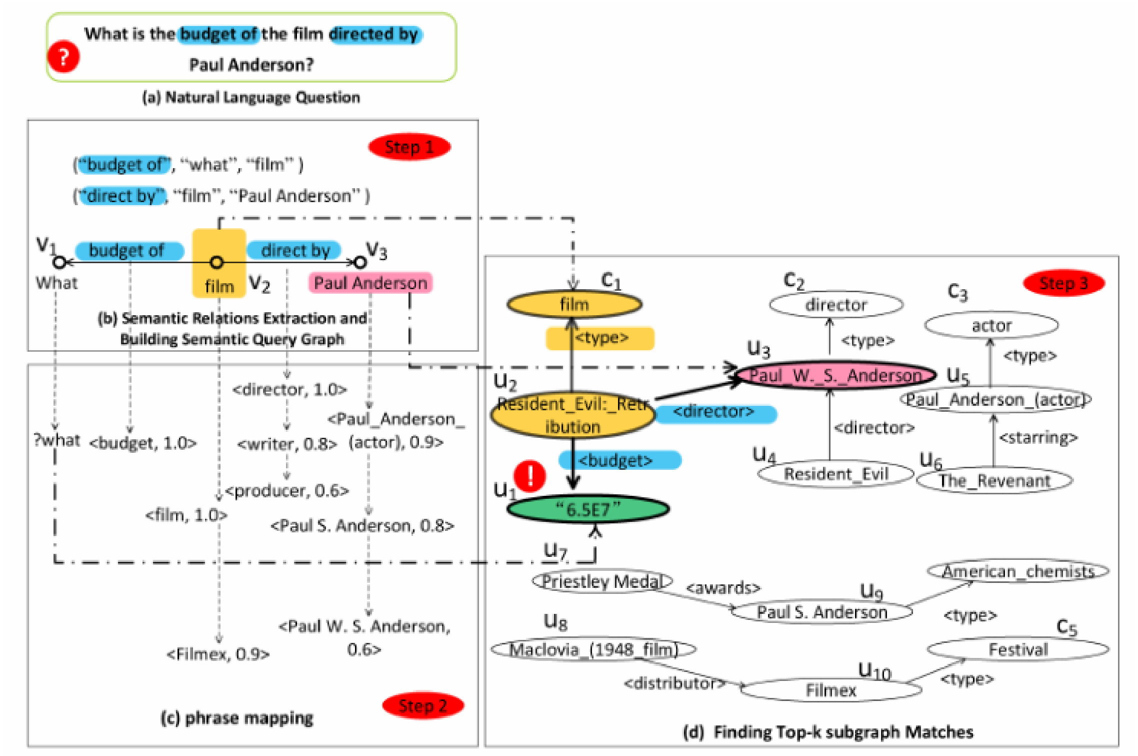
TABLE II: QALD-4 on DBpedia 3.9

	Proceed	Right	Partially	Recall	Precision	F-measure
Xser	40	34	6	0.71	0.72	0.72
gAnswer	25	16	4	0.37	0.37	0.37
CASIA	26	15	4	0.40	0.32	0.36
Intui3	33	10	4	0.25	0.23	0.24
ISOFT	28	10	3	0.26	0.21	0.23
RO_FII	50	6	0	0.12	0.12	0.12
Svega	48	35	6	**0.76**	**0.76**	**0.76**

Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs

[19]

The second paper (Yavuz et. al 2016) takes a slightly different approach, and beings by constructing graphs containing ambiguous entities, and uses subgraph matching to find entities over the underlying Knowledge Graph. This is done through different approaches- Relation-first Semantic Query Graph, and a Node-first Super Semantic Query Graph.


What is the budget of the film directed by Paul Anderson?
(a) Natural Language Question
Step 1
("budget of", "what", "film")
("direct by", "film", "Paul Anderson")
(b) Semantic Relations Extraction and Building Semantic Query Graph
(c) phrase mapping
Step 2
(d) Finding Top-k subgraph Matches
Step 3

Question Answering with Semantic Query Graph in Relation-first framework

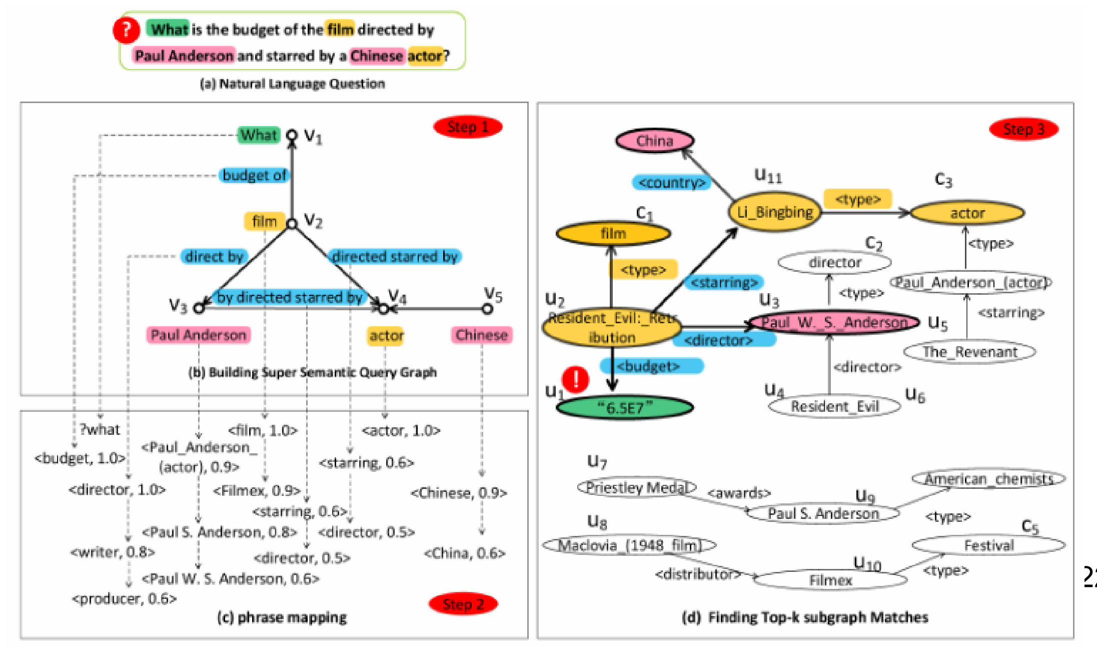
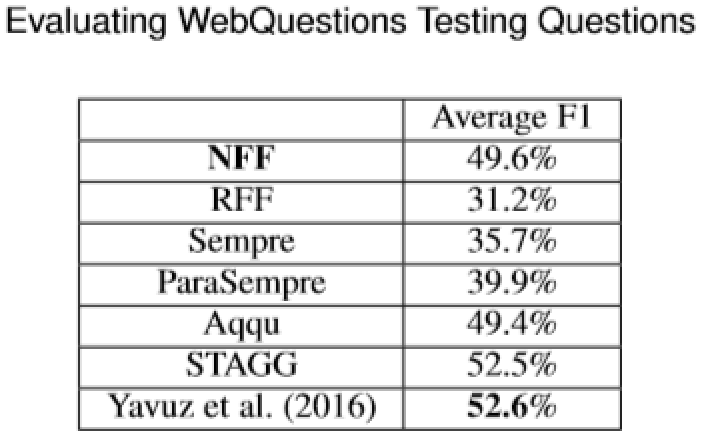

What is the budget of the film directed by Paul Anderson and starred by a Chinese actor?
(a) Natural Language Question
(b) Building Super Semantic Query Graph
(c) phrase mapping
(d) Finding Top-k subgraph Matches

Fig. 3. Question Answering with Super Semantic Query Graph in Node-first Framework

Both of these papers try to solve the main problem of disambiguation. The first paper does so through extracting ***(object,relationship,subject)*** triples from the query, and adding hidden vertices where necessary. The second paper constructs a semantic query graph, either with respect to nodes first, or relationships first. They then use subgraph matching to try and return a result through a structural comparison between the constructed graph, and the underlying knowledge base.

Entity disambiguation seems to be the biggest problem to solve when it comes to querying a graph database, and the current challenge is to find scalable algorithms that can be used to process knowledge graphs of ever-growing sizes.

An important point to note is the results of both these studies are still quite immature, whilst only granting a roughly 45%-55% complete match accuracy rate on certain datasets. While this is impressive for a generalised Question/Answer query of a graph database, being the current state of the art, it is still far from mature enough to be used in a production environment.


Evaluating WebQuestions Testing Questions

	Average F1
NFF	49.6%
RFF	31.2%
Sempre	35.7%
ParaSempre	39.9%
Aqqu	49.4%
STAGG	52.5%
Yavuz et al. (2016)	**52.6%**

As of yet, it is still unclear what direction this project will take, as that will be fully realized after my time here as an intern has elapsed.

My overall takeaway

from my participation in this project was expanding the ways in which I could visualize data. Thinking about data in a myriad of different ways can present solutions that prove to be highly effective.

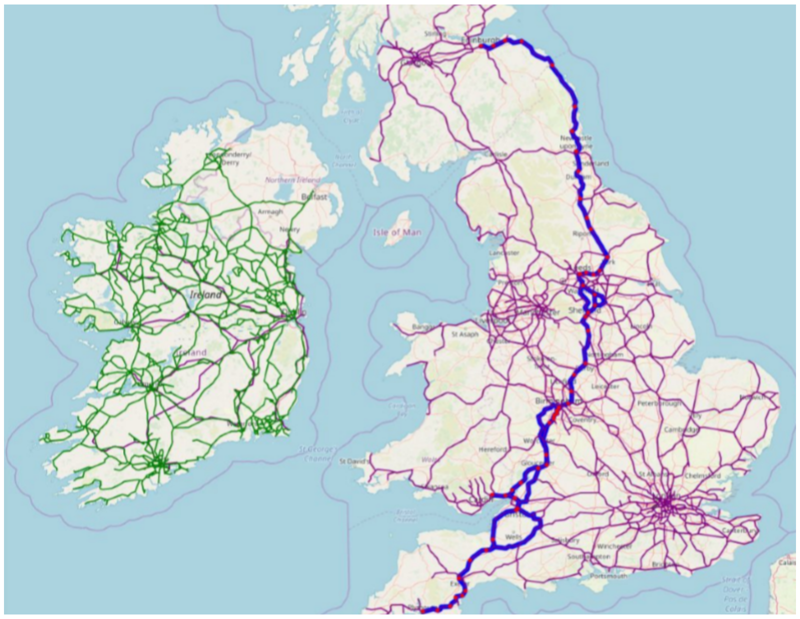
The SkipGram models such as Word2Vec combined with MS-LSTM, mediates thinking about words and concepts as points on n-dimensional vector-space, something which I had never even thought of before.
Models such as those presented in the two papers demonstrates the different ways that a structural-graph-based way of thinking of the problem can also yield state-of-the-art results.

My involvement in this project has enabled me to expand the classes of frameworks that I am able to think about data. The project is also a key example of a problem possible to solve with Machine Learning, but also possible to solve with traditional algorithms if framed in the right way.

Buzmap

Towards the last couple of weeks of my internship, I was preparing material and Quality Testing a product of an acquaintance of mine, from the United Kingdom, that may have been of interest to MMAIS, as a side project. My acquaintance has been in the software engineering industry for over 30 years, and was looking to bring his new product, Buzmap, into the market, and looking for a party to collaborate with.

A brief summary of what my acquaintance has accomplished is a highly scalable vector-tile line reduction algorithm that can be used to place the entire globe's public transit systems on a single map. His method is highly performant, and can handle and process obscene amounts of data, sending it back to the client in a reasonable amount of time.



I worked closely with the engineer behind the project, to draw out a proposal/presentation of his new mapping technology, to try and facilitate a cooperation between the two parties.

This endeavour has given me experience in doing business development through being a third-party in a sales pitch of technology to a company. I concluded this chapter by presenting the product to relevant members in my department.

This unexpected mini-project has grown my business skills considerably. I got to experience the tail-end of the product development cycle, where I would be consolidating the key features, as well as use cases, to be used in a sales-pitch to a big company. I was required to analyze the niche market that my department of Microsoft controlled, and demonstrate succinctly how this new product could add value to a part of their business. I also had to demonstrate how I believed that this type of technology could be scaled up to compete with some of the big international map-makers around, such as GoogleMaps. As a result, I learned the importance of product differentiation by demonstrating how this technology could provide a use-case that the big map makers have yet to employ.

I particularly enjoyed this portion of my internship, because it gave me experience that was not directly related to engineering. I believe my career as a Software Engineer will require soft-negotiation-skills such as the ones gained here, and I am looking forward to continue exploring new opportunities where I can grow not only my engineering skills, but also my business analytics and entrepreneurial skills.

Closing Thoughts

Overall, my internship was an invaluable learning experience. I got to experience first-hand working with top-tier employees and experts in the field, as well as receive excellent guidance and direction from my highly skilled coworkers.

I have practiced the skill of doing relevant research to problems at hand, as well as how to be creative in trying to apply novel strategies to fit a specific problem presented by the client.

This internship has brought me through several deep, yet related fields of study: Graph Databases, Machine Learning, and Natural Language Processing. I have also acquired some business analytics experience, through working with the Bank as a client, as well as being a third party in a sales pitch of a technology to the company.

Working in an office has also showed me aspects of office life, culture, and discipline that I would have otherwise not been able to experience without being fully immersed in the daily routine of working at an office.

These lessons have better prepared me for my career as a well-rounded Software Engineer, as I embark on the step in my professional journey.

Not only did I gain invaluable lessons, but working at MMAIS also brought me new friendships, the opportunity to connect with people of a completely different culture, and to be fully immersed in the working environment of a top-tier Software Engineering department in China.

Over the course of the last two months, my Chinese speaking has definitely improved by leaps and bounds, and I am now more confident that I will be able to work in a Bilingual environment, should I require to, in the future.

I am extremely happy to have had this opportunity, most notably the opportunity to be managed by Ye Fei, an excellent CTO and team leader, as well as have received the patient guidance of Zhang Jing, with his top-tier industry leading expertise in the field of Artificial Intelligence and Machine Learning.

I would like to extend my sincerest gratitude to Ye Fei and Zhang Jing, as well as all the great friends I have made at the office along the way. Thank You for making the Summer of 2019 a fun, productive, and eye-opening growth opportunity.

Ethan Ooi
2019

[illegible]

References

[1] **Neo4j for Finance**
https://go.neo4j.com/rs/710-RRC-335/images/Neo4j-in-Financial%20Services-white-paper.pdf

[2] **FeatureTools Feature Extraction**
https://github.com/Featuretools/predict-loan-repayment

[3] **Towards Data Science - Predicting Loan Repayment**
https://towardsdatascience.com/predicting-loan-repayment-5df4e0023e92

[4] **Rogue Behavior Detection**
https://www.sciencedirect.com/science/article/pii/S2352664516300177

[5] **Neo4j Fraud Detection**
https://github.com/neo4j-contrib/gists/blob/master/other/BankFraudDetection.adoc

[6] **Fraud Detection with Graphs**
https://neo4j.com/blog/fraud-detection-using-graph-technology/

[7] **First Party Fraud Detection**
https://neo4j.com/blog/first-party-bank-fraud-detection-graph-databases/

[8] **Regulations and Data Lineage**
https://www.marklogic.com/blog/implementing-data-lineage-financial-firms/

[9] **Graph Analysis and Machine Learning**
https://analyticsanddatasummit.org/wp-content/uploads/2016/12/F1852609100_20170202_Chaf
i-Hornick_-_Combining_Graph_and_ML_Tech_using_R_-_v10.pdf

[10] **Hierarchical Machine Learning Model for Mechanical Property Predictions**
https://www.frontiersin.org/articles/10.3389/fmats.2019.00087/full

[11] **Multi-Sense LSTM for Knowledge Mapping**
https://www.aclweb.org/anthology/D18-1221

[12] **Word2Vec and Skipgram**
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[13] **2nd Place Solution to Loan Default Prediction**
https://github.com/KazukiOnodera/Home-Credit-Default-Risk

[14] **Convolutional 2D Graph Embeddings**
https://arxiv.org/abs/1707.01476

[15] **Expeditious Generation of Knowledge Graph Embeddings**
https://arxiv.org/abs/1803.07828

[16] **Deep Path - Reinforcement Learning Graph Embedding**
https://arxiv.org/abs/1707.06690

[17] **KBGAN - Framework to improve model training**
https://arxiv.org/abs/1711.04071

[18] **Svega - Answering Unstructured Natural Language Questions**
https://pdfs.semanticscholar.org/71a2/eedacd4ed25e6996810908d17bce08862d3c.pdf

[19] **Answering Unstructured Natural Language Questions by Subgraph Matching**
https://ieeexplore.ieee.org/document/8085196