Blue – Interviewer1, Purple – Interviewer2, Green - Interviewee

So just if you could confirm that you have agreed for us to tape the interview and share it's transcription.

Yeah. Yeah, I agree.

Yeah, thank you. So maybe we could start by asking you some background information. So what is your current position?

So I just graduated. I just got my PhD in computer science at the [University: Removed for Anonymity]. I had a bachelor and master in Computer Science in [University: Removed for Anonymity] and I had five years of PhD here at [University: Removed for Anonymity].

Okay. So how many years of overall work experience do you have and specifically in Deep Learning Systems?

Outside of academia or in academia?

If you can just tell each of them.

Okay. So outside academia, in industry only three months of internship at [Organisation: Removed for Anonymity]. And for academia in deep learning probably only two years.

Okay, thank you. So what type of deep learning networks have you developed/implemented and by that I mean supervised/unsupervised/reinforcement learning and so on.

We implemented two types. First, I started with unsupervised with autoencoders, recursive autoencoders for task in software engineering like clone detection. Then we used language models, so Word2Vec. Then, more recently we use supervised model encoder/decoder model, so sequence-to-sequence models.

Okay. So which problems did you try to tackle using these networks? One you have mentioned is clone detection, right?

Yeah, so. Basically, with recursive autoencoders we tackled mainly clone detection and learning similarities from source code.

Okay.

And for when we use RNN encoders/decoders model we tackled different problems such as automatic bug fixing, automated program repair, learning mutants, so mutation testing and learning more general code changes, such as refactoring and etc., a code review system.

Okay, thank you, which programming languages and frameworks have you been using during this development?

So when we started, the autoencoder one is implemented in Matlab and we did not use any framework, just like Matlab code. Okay, for all the most recent projects we are using Tensorflow.

Okay, so thank you. So this is all the background questions that we had. So, as I have mentioned we have one general question for this interview: which is the types of bugs and problems that you have faced. So, maybe we can just start with that general question and you can tell us about the things that you have faced in your experience.

So, the general question is what type of challenges?

Not exactly challenges, but also bugs and problems.

Hmm. So when I started with Matlab code, it was very difficult to test that we were actually learning a good representation. So, testing was the biggest challenge and like we wasted a lot of time, trying to like basic like bugs in basic mathematical operations because there was very little support for training this network, at least in Matlab, at least when I started. For, more recently when I switched to tensorflow, I don't have this kind of bugs anymore. I mean, I use like some sort of like high-level API, so it is easier to not to make mistakes on that. But, so, most of the bugs now are often in the data pre-processing part; in the integration part, like integrating like both the encoder and decoder; in the inference part, like sometimes when we want to perform like some particular types of inference, like not only one output, but multiple outputs, we have to deal with probabilities and sometimes that maybe we need to develop some scripts and sometimes there are some bugs in these scripts, so in the inference part. Other bugs are more related to like hardware level bugs. For example, I don't know if this is related to your study, but basically one of the problems we have is that when we run on GPUs, usually it is not as running on CPUs. So for example, if I run a like WEKA or some other machine learning approach, how will we be able to run shared with other people. Instead, if someone else is already using that GPU, it is very difficult to share that GPU with other users. So you either need to wait or your program will fail or something like that. And then yeah in terms of bugs, that is it. Other challenges for are like the fact that I need to deal with a lot of different files. Like I don't I don't have like a single IDE that can control everything. For example, I have to deal with Python's files, but also running files, like sh files, configuration files like yml. Then I have to open different tools like TensorBoard. So all dealing with those this type of files without any single like IDE integration.

Okay, anything else you want to add?

No, thank you.

So you mentioned that you have you deal with pre-processing bugs. Could you give more details about that?

Yeah, so like a very stupid one was that we were using, we're doing part of the pre-processing by ourselves. And then we were feeding to another model that was doing again another small part of the pre-processing. So, in our pre-processing we were using tokens like unknown tokens, UNK token, and this one was reserved for the other approach. So this was giving a weird bug for us. Other bugs are like all related to the vocabulary sides or like special characters. Yeah, mostly these types of bugs.

So vocabulary size, do you mean you get problems when their size is very big? And how do you deal with that?

Well, we usually what we did is like we try to reason how to limit this vocabulary. We first try to act like just set a threshold and then we try to just change the entire representation using like replacing some IDs with other ID. Having a higher level of abstraction in the vocabulary.

So, integration bugs that you mentioned. What are you integrating and what are the…

So they're not very difficult bugs. So basically, sometimes we try to, so this infrastructure, this TensorFlow infrastructure allowed us to use different encoders and different decoders. And sometimes the configuration of the encoder was not compatible with the configuration of the decoder. So we were running in some trouble, but that was mostly based on our like limited understanding of the infrastructure.

So, can I ask a question? What do you mean by configuration there?

Yeah, so like for example in Tensorflow you can define some yml files to define the model. And basically you can define like the embedding sides, the type of encoder and the type of training steps and etc., activation function and etc. So one bug was like we set a different embedding sites for the encoder, it was incompatible with the one expected from the decoder. So this was a very stupid bug that we saw. This I mean are like high-level delegation, not very specific.

Okay, and you mentioned inference problems. Any details about that?

Yeah, so. We had to change the inference of our model. So instead of like basically asking only one output, we wanted like the top-K most likely sentences. So we had to like dump the probabilities and then try to create our own script to obtain the most likely sentences and these was just like some sort of bugs on how to interpret those probabilities. But it is always related to like our early understanding of the outputs. So, we were not sure of what we were getting as output and then we had to like dig into what was this output and then to fully understand what was going on. So I felt like there was like limited documentation for that particular output. But yeah, it's not like a fault of the infrastructure, like it is more limited to a to us.

Okay, I see. Okay. Thank you. So when training your networks did you use existing datasets or did you collect your data yourself?

No, we collect our own data.

And did you do it automatically or did you have to do any manual labeling and etc.?

Mostly, mostly automatic. Yeah, so I am involved also in a project where we have these like manual labeling, but it is done on Mechanical Turk. Okay, it is not really manual for us.

Okay. One question, when you crowdsource it, how do you evaluate the submission of people?

So, we had to do some manual work here, we had to filter out a lot of bad labels and it took like a long time. I didn't do it myself, but I know the experience of these students that have done this.

Okay. So, do you have like any number on mind, how often crowdsourcing people mislabel things?

No, I don't remember an exact percentage, but it was not a limited one. It was not just like very small one. We had to throw away a lot of data.

Okay, so do you remember any kind of problems related to training data that you faced in your development? Except the ones that you've already mentioned.

Yeah, no. [Pause] It's always mostly related to like how we pre-process the sentences and the source code and natural language before we feed to the models.

So to pre-process the source code, so do I understand correctly that you use some kind of language parser?

Yeah. Yeah. So we use like a lexer and parser.

Okay.

We then represent this source code like as a string, a sequence of tokens and like we do some renaming and we insert special tokens, things like this. Yes.

Okay, and this process like usually goes smoothly. You don't get into any trouble while doing that?

So, yeah, definitely we had like bugs related with this. But yeah, it's like is mostly like on the implementation of how to pre-process this source code and early on, before we even feed to the network, so. So yeah, definitely implementing this lexer and parser infrastructure we had to do a lot of testing and we had a lot of bugs, as it happens often.

Any bug on your mind that you could mention?

So, they are very specific to our implementation, like [Pause].

It's okay, if you don't remember, we can skip that.

Yeah, so it was like, we couldn't we didn't catch some programming construct. The lexer and parser were failing on this programming constructs and we were not like replacing correctly the same tokens. So yeah, this was mostly the problem.

Did you always catch this type of bugs before you even started the training? Or did it ever happen that you trained, you saw that it doesn't work for some constructs and you went back to your training data?

So the case where we had to go back was where we saw the unknown token. So, we were using some special tokens on our end and then these were incompatible with the existing toolchain in the Tensorflow. So this was immediately a big problem for us, like because we saw the training stopping, like we couldn't figure out why. Then, other things that we went back was like mostly like or we train and we see that we overfit pretty early. So we go back and try to change some data pre-processing step and see what's going on. But yeah, the main bug that was like related to like not being able to train was only these special characters. Yeah.

Okay. Okay. Thank you. So about model structure. So you said that you use different kinds of models. Do you decide on their structure yourself or do you use existing ones? And do you have problems related to wrong model structure, such as number/types of layers. Some dimensions and etc.

So, before we start, we usually go check in the literature and see what are the suggested architectural models. So, we didn't, we never had problems in like finding the right infrastructure and model. But, we definitely had to like experiment a lot with like the type of activation function and number of layers. Like for example, when we started, we were thinking that we needed at least four layers in the encoder and the decoder and then we ended up like having half of them, like actually very very shallow model and was even better than the biggest deepest model. So this was like something some assumption that we made and like actually at the end when we did the hyperparameter search, we ended up being surprised that we actually needed less layers and less neurons.

Okay, so you mentioned hyperparameters and I would like to ask if you had any problems related to hyper parameter tuning?

Problem. So not really problems. We were just like just trying to find like the optimal strategy to find these hyperparameters. We because we know that it took a lot of time to train a single model. So we were like kind of struggling to like find what are the right combination of hyperparameters that we should test. We went to literature and see like okay, let me see what they other people user usually use. So to reduce our search space, but definitely we had to like make some our own assumptions say, okay, maybe we can try, maybe we have like 20 trials that we can do and let's try to do check change this. We didn't use like randomizers, because we didn't have enough trials so we try to do like some like very sparse grid search, but no particular problems. Like other than like maybe once we set too large number of neurons and we had like very slow training and validation, so we basically just kill that process. This was the main problem, but no particular bug.

Okay, thank you. And about loss functions, do use the predefined or a custom-written loss function? Did you have any problems because of the loss function?

No, so usually we use predefined loss functions. We never customize the loss function.

Okay.

We definitely need to like, what we do mostly is like keep track of the loss function like both on training and validation set. Just make sure we don't overfit. And like we had some thinking. Basically when we realize that we have two ways to evaluate our score at the end. So we were thinking whether to change or not the loss function because we found that was not a good proxy for us at the end.

Okay.

But but yeah, we didn't we didn't change loss function at the end. We are still thinking how what is the best way to do it?

But you do have a loss functions that you think that it's not a good proxy, right?

So, I have like some clue that might be we might be better using a different one, but. But yeah, I don't have a definitive answer on this, and it might be just like my impression.

Okay, so maybe during our conversation you remember some bugs and problems that you faced. Maybe you have anything to add?

Even small technical problems or issues. That would be useful.

[Long Pause]. So, I mean there are a lot of like bugs like reference to checkpoints that don't exist anymore, like when we want to retrain a model and we lose the checkpoint, a lot of very easy bugs related to path to files. Yeah, mostly mostly this type of bugs.

Yeah, okay. So you told us that you face some problems related to hardware and that's sometimes really hard to share the same resources. Maybe there are some other issues if tried some concurrent running of your programs. Maybe there are some issues like race conditions or something like this?

Yeah. No, I don't think was race condition problem. It was mostly that, like we use a server that is shared by a lot of other students alike. Basically, we don't have like a queue system, where we can just submit a job and like wait for our job to be performed. It was like a live session. So a lot of times, like someone is using this GPU and we cannot like run our model, so we have to wait for this. So this is like a very, mostly based on how we have this server set up. So we didn't have this problem like when we used the clusters where we can just submit a job and just wait for it.

Yeah, thank you. I wanted to ask about the code clone detection thing that you mentioned, just to understand the process. So you parse different kinds of code. Do you, is it like, do you label them as being close to each other, or is it rather unsupervised? And what is the output of the machine learning or deep learning systems that you expect?

Okay, so just to make sure when I talked about lexer and parser, I was talking about RNN encoder/decoder, the one that is supervised and the one that is related to automatic bug fixing and etc. For the code clone detection, what we do is like we still parse both. Basically it is unsupervised in the way that like we start with this code representation. We don't label two pieces of source code to be clones. What can we do is like, we use a recursive autoencoder to generate a representation of this source code. Like imagine like Word2Vec, but not at token level, but a piece of source code. So and then we use like, so the model is trained to generate this representation, in such a way that minimizes the reconstruction error to reconstruct the original piece of source code. So, then what we do is just like we compute the similarities of true embeddings and see if they are similar, if they are not very far away, we assume that they are close, because their representation is close enough. But, yes, it is unsupervised because we don't use labels.

So, is the output of the system binary? So it either says it is a clone or not?

No, it is a distance. This is a similarity score.

Okay.

So yeah, we use cosin similarity or Euclidean distance.

And you put a threshold on this distance?

Yeah. Okay.

So could you tell us a bit more how you decide on this threshold?

Okay. Yeah. So, this threshold is definitely like a, it is a problem of a lot of clone detectors that use thresholds Is kind of like a decision we have to make. So what we did is like based on the programs, we observe the distribution of the of the distances, for all the for the pairs and try to come up with an empirical threshold that works. Andthen we like we set one so, it's not we did not like learn this threshold and we did not like do any like in-depth study on this.

Okay. Okay, so we don't have any other questions. If you have anything to add you are welcome.

No, I think that's it.

Okay, so, thanks a lot again.

Thank you.

Thank you. Bye. Bye.