CS230

Graph-Convolutional Neural Networks for Hydrocarbons Kinetics Predictions

Vincent Dufour-Décieux
Department of Materials Science and Engineering
Stanford University
vdufourd@stanford.edu

Anthony Taing
Department of Computer Science
Stanford University-SCPD
anth248@stanford.edu

1 Introduction

We propose to predict the kinetics of the pyrolysis of hydrocarbons using Graph Convolutional Neural Networks. Pyrolysis of hydrocarbons is a complex chemical process involving thousands of different molecules and of different reactions, making it hard for a simple model to predict its time evolution. Commonly, time evolution is predicted using simulations called Molecular Dynamics (MD), but these simulations involve computing the movement of each atom and their interactions with each other, making this process computationally expensive. Recently, data-driven approaches using the output of these simulations have been developed to accelerate the prediction of the evolution of the system. These data-driven methods use techniques such as Kinetic Monte Carlo [1], PDE-approximation [2], or Machine Learning [3]. However, to our knowledge, no work has been done using the powerful framework of Graph Convolutional Neural Networks (GCNN) to predict chemical kinetics. Indeed, this graphical approach is particularly well suited for describing molecules and crystals, which has resulted in growing interest in the chemistry community over the last few years [4, 5, 6].

In this work, we present a framework coupling GCNN and Kinetic Monte Carlo (KMC) in order to predict the evolution of a chemical system. Given the reaction rate at which each pair of atoms in the system reacts, the KMC algorithm picks the next reaction that will occur and the time before this reaction to occur. For more details on the Kinetic Monte Carlo, the reader can refer to Ref. [7]. The purpose of the GCNN is to give an estimation of the reaction rate for each pair of atoms, depending on features related to each atom, and features related to the relation between the two atoms of each pair. To know if our model performs well, we will compare the results of the evoltution of the system using the GCNN/KMC framework with the real data obtained with the MD simulation.

2 Dataset

The dataset is the output of MD simulations, obtained by Vincent through his work in Prof. Evan Reed's lab at Stanford University. It is possible to obtain the connectivity of the networks of atoms to create a graph of the system from the positions of the atoms at each timestep. So far, we have 3 simulations of around a thousand atoms for 50,000 timesteps, with each simulation starting with a different hydrocarbon, but we used only one of them for the training. From the connectivity of the networks, the reactions that occur during the simulation can be obtained. A reaction is defined as the creation or the breaking of a bond between 2 atoms, which can be seen as the addition or removal of an edge in a graph representation. During one timestep, there can be any number of reactions happening (e. g. in our simulation this number goes from 0 to 8). The goal of this work is to estimate the probability that a pair of atoms reacts. However, in most cases no reaction occurs, which biases the dataset towards negative data (no reaction). There are around 50e9 possible reactions (= 1,000*1,000*50,000) however there are only 26,000 that actually occur, which is less one reaction

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

for one million possible reactions. In order to deal with this issue, we decided to give the constraint that exactly one reaction would occur for each data point. Several rationale explain this choice: first, it is a natural way to couple GCNN with KMC, since the KMC algorithm only picks one reaction to occur. Secondly, first trials (reported in the milestone) showed that if the model does not have the constraint to pick one reaction, it will give reaction rates very close to zero for each pair of atoms due to the highly biased data, and it will not choose one reaction over the others giving a behavior close to random. In addition, the model tries to optimize the loss of the system and if the loss is compared with cases where there are mostly no reaction, the model will just try to go to zero without paying attention to which reaction is more likely to react. Therefore, the loss of the model is not related to how well the model performs, making it hard to train. In the case where exactly one reaction occurs for each data point, the model will have to order reaction rates and using a softmax as the last activation layer will force the sum of the outputs to be exactly one. In order not to lose the time component of the initial MD simulation, we also report the time before the next reaction to occur. If several reactions occur during one timeframe, a random order is chosen and each reaction is separated by the time between two timeframes of the MD simulation. Thus, the model needs to predict what is the probability that each reaction is the next reaction to occur and the time before this reaction. So for each data point, there are two outputs: the first one is an array that has the number of possible pairs elements where all of them are 0, except for the unique pair of element that is 1. The second output is a single number representing the time before this reaction to occur.

The inputs of the model are a construction of the graph representing the chemical system, in addition to features for each atom and each pair of atoms. Each atom has three features associated with it: its type (carbon or hydrogen), the size of the molecule it is in, the size of the cycle it is in (and 0, if this atom is not part of any cycle). Therefore one of the input is an array containing the three features for each atom at each timestep. Then each pair of atom has three features associated with it: if the atoms are connected (1), or not (0), the distance between the two atoms if they are in the same molecule and 0 if they are not, and if they are part of the same cycle (1) or not (0). The second input is an array containing the features for each pair of atoms at each timestep. Two other inputs help to create the graph in the model, giving some connectivity information: the first one is an array giving the list of atoms bonded to each single atom. The second one is a mask in order to avoid counting bonds that do not exit. Finally, a last input gives an index to each possible pair of atoms in order to remove duplicates.

All these data are quite heavy but also sparse, therefore they are stored in an HDF5 file with each dataset containing the sparse array corresponding to one frame. The data are then read using a Python generator, because only few hundreds frames can be loaded at once due to their memory weight (the array containing the features of the pairs of atoms has an approximate of shape (num of examples, 1,000, 1,000, 3) which is quite big).

3 Implementation and Exploration

For the model, we use a GCNN inspired by an implementation by Coley et al. [5] shown in Figure 1. Using the data described in Section 2, we set up the graph, with nodes representing atoms and edges representing bonds. The system goes through different steps each represented by a GCNN. The first one is a loop encoding the description of each atom. The features of each atom is encoded using its initial features but also the features of the atoms it is bonded to. We did not exactly followed the architecture presented in Figure 1. Out of the loop, the features of each atom were encoded using three hidden layers containing 64, 64 and 16 neurons each. Then, inside the loop, the features of the neighbors of each atom where encoded individually and then summed. The summation here is to preserve the symmetry between each neighbor: indeed, we don't want to concatenate here since it would give a different role for each depending on the order these neighbors are presented. Then, the summation of the features of the neighbors goes through a ResNet layer and are then concatenated with the features of the atom. Going through this loop several times allow to obtain more precise information. After running, some tests, going through the loop 5 times gave the best results, even if this number did not have a big impact on the results. The number of layers for each part has been tested and three layers gave the best results. The number of neurons were increased but due to the big number of data in each data point, we couldn't increase the number of neurons without obtaining an "out-of-memory" error. The ResNet layers were tested and gave a slight improvement so they were kept.

The features of each atom were then paired by multiplication for each pair of atom. The multiplication here was once again in order to keep the symmetry of the system. We will call these features the "pair of atoms features", in order not to mistake them with the input presented earlier that is the "features of each pair of atoms". The difference between these is that the former is the coupling of the features associated with each individual atom, whereas the latter represents the features of the pair of atoms (e. g. are the two atoms of the pairs bonded ? In the same molecule ? In the same cycle ?). Then the "pair of atoms features" and the "features of each pair of atoms" go through different neural networks, both composed of 3 layers. The results are then concatenated and go through a new neural network of three layers. All these layers follow a ResNet framework that showed good performance, and the number of layers were tested too (we tested 1, 3 and 6 layers) and 3 layers gave the best performance.

Each pair of atoms final features goes through a hidden layer that gives a single score as an output. These scores go through a softmax layer, to give a prediction for the next reaction to occur. The exponential of all these scores were then summed in order to predict the time before the next reaction. This summation is a natural way to predict the time before the next reaction, because this is how the time before the next reaction is chosen in the KMC framework. Finally, we calculated two losses: the first is a categorical cross entropy that compares the probability that each reaction occurs with the reaction that actually occurred. The second one is a MSE that compares the prediction of the time before the next reaction with the actual time before the next reaction.

4 Hyperparameters Tuning Experiments

An Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used. We used the keras.callbacks.ReduceLROnPlateau to help us schedule our learningrate. The learning rate was chosen to be 0.001, and was divided by 10 every time the loss didn't decrease by more than 0.1 for 3 epochs in a row.

This architecture was the results of many trials. The trials were performed on 2,000 to 5,000 data for 5 to 10 epochs. The validation set could not be bigger than 10 without creating "out-of-memory" errors, therefore the training loss was used for comparison. As we will show later, the results obtained never seemed to be overfitting; therefore, the training loss was a good metric to know if our model was improving. In addition, since we never overfitted, no regularization was used (dropout, L1 or L2 regularization) except on the last Dense layer. Indeed, having regularization there helps to have a low final score, which helped the predicted time to be in the good range (the time values were on the order of 0.01). As mentioned before, we explored the number of layers for each encoding and their number of neurons, the earlier was defined as 3 layers for each encoding, and between 16 and 64 neurons for each layer. No test above 64 neurons was performed because it gave out of memory errors due to the amount of data. Xavier, He Normal and Le Cun Normal initialization were tested for each layer, and Le Cun gave the best results and was used here. ReLU, SeLU, Swish and Tanh were used as activation functions. SeLU and Tanh gave the best results and Tanh is used in the results shown later. Different learning rates were studied (from 0.01 to 0.0001) and different schedules were studied, and a learning rate of 0.001 with the schedules given above gave the best results.

For the losses, MSE, MAE and binary cross entropy were also tested for the loss trying to predict what is the next reaction but none of them showed good results as we expected since the categorical cross entropy coupled with a softmax activation on the last layer seemed appropriate for our problem. Adam, Adagram, Sgd, RMSProp were also tested for the optimizer and Adam was the best optimizer. We also considered to use Batch Normalization but it gave bad results. It can be explained by the fact that we could not increase the batch size to more than 10 without having "out-of-memory" errors due to the size of each data point. Batch Normalization only works well when the batch size is big enough which was not the case here. Finally, a last hyperparameter that was studied was loss weight for the two different losses we considered. The loss weight for the loss associated with predicting the next reaction was fixed to 1 and the loss weight of the second loss was varied. What was found was that giving a very low value to this number gave the best result. Indeed, putting a L2 regularization on the last layer gave a better loss that letting the second loss being optimized by itself.

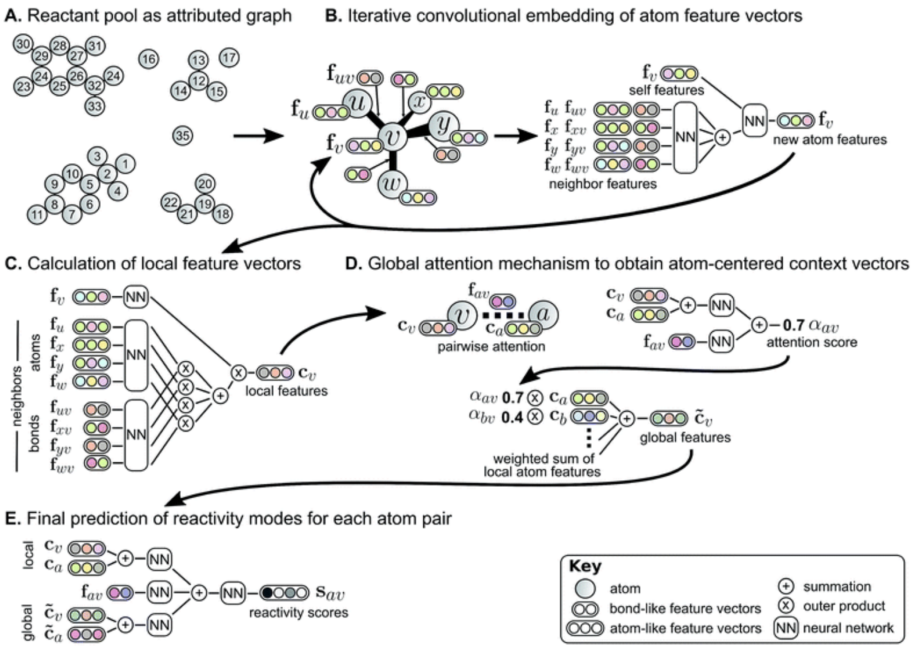

Figure 1: General framework used by Coley et al. [5] to predict the reactivity score of each pair of atoms. In this work we did not use the step D. because it is only important if the nature of the atoms present in the system changes, which is not the case here (only carbon and hydrogen).

Loss: categorical_crossentropy	
Activations+initializer	Final loss value
Relu+he_normal	5.8
Selu	5.5
Tanh	5.5
Tanh+he_normal	5.8

Loss: categorical_crossentropy and optimizer: Adam	
Hidden layer size	Final loss value
16	7.7
24	6.0
32	5.8

Figure 2: Final loss value by activations, initializers and by hidden layer size.

5 Results

For the results presented here, we trained the model presented earlier on 25,000 data points for around 15 epochs. The final training loss was around 5.8 and is almost only due to the loss of the prediction of the next reaction. To give an order of idea, a random guessing loss would be a bit more than 12. This corresponds to an accuracy of 13% for predicting the next reaction. The results on the validation were slightly better but as explained earlier. Since there were on only 10 data points they were not statistically significant. All these metrics seem to give that the models did not perform well. However, the goal of this work is to couple the prediction of the GCNN with a KMC framework. In Fig. 3, we can see the evolution of the 2 molecules that are present in the highest number for the MD simulations, C_4H_{10} and CH_4, in addition to the longest molecule. These are considered as good descriptors of the simulation. The goal is to have the "GCNN/KMC" curves as close as possible to the "MD results" curves. The time is normalized on all the figures because the time prediction were not good and would give bad visualization. All the curves are the results after 300 reactions. As we can observe the "GCNN/KMC" are not matching the "MD results" curves. Some details of the results of the tuning are shown in Figure 2.

6 Discussion

Unfortunately, we were not able to achieve good performance in this project. The GCNN/KMC was not able to reproduce or even get close to the data of the MD simulations. Not only the curves don't look great, but a deeper analysis shows that the model doesn't understand the chemistry of the system at all. For example, in chemistry it is known that a carbon is reactive if it is not bonded by 4 atoms.

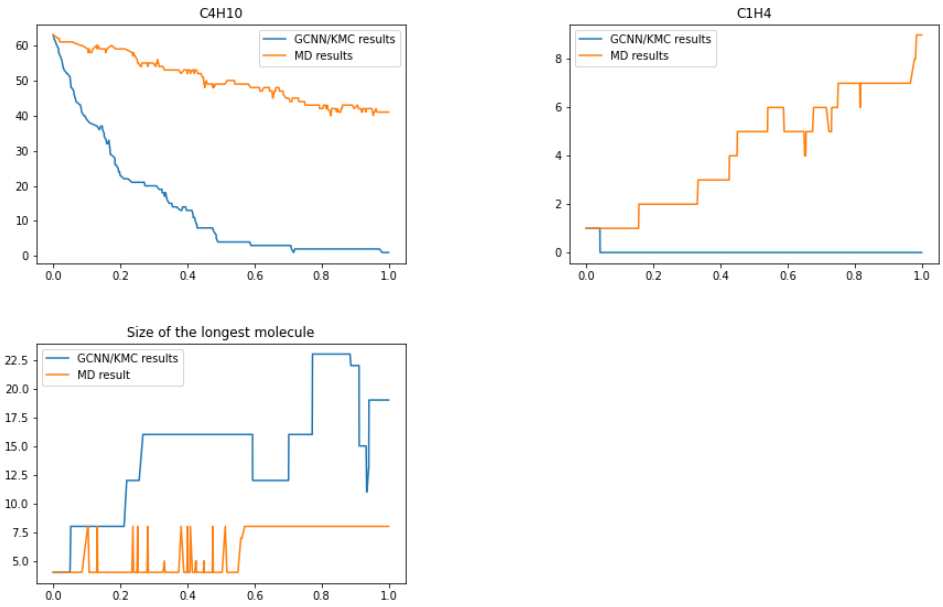

Figure 3: Evolution of the number of molecules C_4H_{10}, CH_4 and the size for the longest molecule of the system versus normalized time.

And the further it gets from 4 neighbors, the more reactive it becomes. However, it does not appear in the results. For example, some carbons reach 1 or 9 atoms in the GCNN/KMC system, which should be impossible, and they are not the most likely reaction to occur. So it seems like the model doesn't understand some basic chemistry rules.

Several reasons can explain these bad results. First, as mentioned several times above our dataset was very big. Each input was big especially the features of the pairs of atoms that had a shape of the order 1,000*1,000*3 for each data point. This limited the number of layers and the number of neurons per layer, so our model ended up having only 23,000 parameters, which doesn't seem that much. A solution would have been to reduce the size of the system, but this would also reduce the number of observed reactions. However, for future work, this could be an option. Another reason could be an inappropriate architecture. The chosen architecture is close to some previous work by Coley et al. [5] which did some similar work and his architecture seemed natural from a chemistry point of view. However, other architectures could be tested in future work.

7 Conclusions

To conclude, although we tried to adapt our model to overcome the low number of reactions, and the heaviness of each data point, we were not able to produce satisfying results.

8 Contributions

All team members contributed to the project through weekly meetings, frequent check-ins and contributed to proposal writing, milestone writing and final report writing. Vincent conducted data manipulation and preprocessing. Anthony conducted the data storage and set up the AWS environment.Both contributed to set up the model, run experiments, and performed hyperparameters tuning of the model.

Github repository :https://github.com/Spyciblock/hydrocarbon_kinetics-predictions.

References

[1] Qian Yang, Carlos A Sing-Long, and Evan J Reed. Learning reduced kinetic monte carlo models of complex chemistry from molecular dynamics. *Chemical science*, 8(8):5781–5796, 2017.

[2] Yanze Wu, Huai Sun, Liang Wu, and Joshua D Deetz. Extracting the mechanisms and kinetic models of complex reactions from atomistic simulation data. *Journal of computational chemistry*, 40(16):1586–1592, 2019.

[3] Jinzhe Zeng, Linfeng Zhang, Han Wang, and Tong Zhu. Explore the chemical space of linear alkanes pyrolysis via deep potential generator. 2020.

[4] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[5] Connor W Coley, Regina Barzilay, Tommi S Jaakkola, William H Green, and Klavs F Jensen. Prediction of organic reaction outcomes using machine learning. *ACS central science*, 3(5):434–443, 2017.

[6] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[7] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.