OrgoNet: Organic Chemistry Reaction Prediction with Machine Learning

Ethan Chi
Dept. of Computer Science
Stanford University
ethanchi@cs.stanford.edu

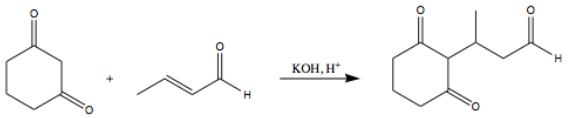
Bowen Jing
Dept. of Computer Science
Stanford University
bjing@cs.stanford.edu

Emily Wen
Dept. of Computer Science
Stanford University
ewen22@stanford.edu

1 Introduction

A major challenge in organic chemistry is predicting the products of the chemical reaction of two molecules. Although simple reactions can be unambiguously predicted using common heuristics, more complex reactions, including those of interest in the pharmaceutical and biochemical domains, are challenging; indeed, the most complex and cutting-edge reactions are typically analyzed by performing them in the lab, then analyzing the results. This is an expensive and time-consuming process. In this paper, we present several graph-based machine learning methods to identify the most probable product of a given reaction.

Typical organic chemistry reactions involve two sets of molecules: *substrates* (molecules that participate in the actual reaction) and *reagents* (molecules that act as 'catalysts', necessary for the reaction to proceed but that do not participate in the final product). For example, in this reaction, the substrates are the left two molecules, the reagents are {KOH, H^+}, and the product is the right molecule.



The majority of synthesis problems of interest follow the following pattern: given some set of reagents, two substrates bond to form a single molecule. Most models take an end-to-end approach: taking in two substrates and the set of reagents to predict the structure of the final molecule. However, since most atoms in both molecules are irrelevant to the bonding process, it is much less computationally expensive to predict which atom from each molecule will form a bond (the 'active atom'). We could then later easily identify the most likely output molecule by using well-understood chemical heuristics. The model we propose here uses such an approach on bimolecular reactions. We take in a graphical representation of two molecules, compute a vector embedding for each atom, and predict the most likely bond to form between the two molecules.

2 Related work

Generally speaking, the state of the art in reaction prediction has been the use of *reaction templates*, molecular subgraph patterns that can be learned for each reaction. Since multiple templates might be identified for a given set of reactants, a supervised model is trained to identify the most likely pairing. For example, Coley [1] implements this method using templates described in Law [2].

However, there are two major drawbacks to this approach. Firstly, in order to match all possible pairs of reactants, an extremely large number of templates is necessary. Although there are heuristics to find templates, most current systems utilize human knowledge. Therefore, the reaction template system is generally not scaleable.

Jin [3] use a Weisfeiler-Lehman Network, a graph kernel model based on graph isomorphisms, to rank possible pairs of atoms. This is extremely computationally expensive, requiring several attention layers to achieve good results; however, it achieves good accuracy due to being able to integrate information from other molecules.

Fooshee [4] uses recurrent neural networks to predict electron sources and sinks of elementary reactions. While this model is more generalizable due to its size and the linear nature of its input features, its use of SMILES strings, a linear representation of molecules, means that it cannot efficiently take into account the graphical nature of the molecules.

Segler [5] uses deep neural networks for reaction prediction and retrosynthesis, where the aim is to predict the reactants of a reaction from its product. This method is entirely based on existing reaction rules. Hence it is dependent on traditional human discoveries, and is thus not extensible to new reaction rules. However, it is effective at applying and prioritizing existing rules.

Schwaller [6] uses a multi-head attention Molecular Transformer model to predict the products given reactants and reagents. While it is an effective model and is quite applicable, it is unnecessarily complicated and computationally expensive.

3 Dataset and Features

We use reactions extracted from the US Patent and Trademark Office [7]. Each reaction consists of a set of substrates, a set of products, and a reagent (incl. 'no reagent'). We restrict our model to bimolecular reactions because they most easily permit the identification of a single 'active atom.' Each molecule is encoded as a linear representation of its graphical structure (SMILE). In preprocessing, we identify the 'electron source' atom and 'electron sink' atom (see below) for each reaction using several organic chemical heuristics.[1].

Our dataset has 185,331 reactions, but we currently use a truncated set of 15,000 reactions for faster iteration. Due to the setup of our model, we restrict ourselves to reagents that have at least 50 reactions. This restricts us to 10 reagents and 10,647 reactions. We used an 80/10/10 train/val/test split.

Our models generate a *vector embedding* for each atom based on its context. We then generate a dataset composed of every possible ordered pair consisting of one atom from each molecule (of size $2 * n_{\text{source}} * n_{\text{sink}}$). Each such pair is labeled 0, except for the correct source-sink pair, which is labeled 1. This strategy allows us to produce a large number of negative examples. However, since the number of positive examples is limited, we have a significant class imbalance problem, which we handle by weighting our loss function.

4 Methods

We further simplify the problem by considering the fact that in almost all cases, a bond forms between a *source*, which donates electron density, and a *sink*, which receives electron density. Active source atoms are likely to be electron-rich (highly electronegative, such as oxygen and nitrogen), while active sink atoms are likely to be electron-poor (near electronegative atoms

Therefore, our problem statement is as follows: given two substrates graphs, represented as sets of nodes G_1, G_2, and a reagent R, we want to predict the source and sink. To do so, for a potential source-sink pair $(a_i, a_j) \in \{(G_1 \times G_2) \cup (G_2 \times G_1)\}$, we predict the probability $P((a_i, a_j), R)$ that (a_i, a_j) is the correct pair. Then, for a given pair of molecules, we predict $\arg\max_{(a_i,a_j)} P((a_i, a_j), R)$ as the bonding pair.

In order to train the models on graphs, we must vectorize each node of the graph. In our logistic regression and Gaussian discriminant analysis models, we vectorize using a simple weighted count; specifically, we embed each node of the graph using a sum of all one-hot-encoded atom identities in the graph, weighted by distance from the current node. This has the effect of encoding an atom based on both itself and its neighbors. In our convolutional graphical network model, we feed in the atomic identity of the node to the network and the network learns an embedding.

4.1 Logistic Regression

In our logistic regression model, we model the probability of a bond forming between a source atom a_i and a sink atom a_j under the reagent k as the following:

[1] In order: atom electronegativity, the presence of nucleophilic aromatic substitution, the presence of electrophilic aromatic substitution, and neighboring electronegative atoms

$$P((a_i, a_j), R_k) = g(x_i^{\mathrm{T}} A_k x^j)$$

where g is the sigmoid function, $x_i, x_j \in \mathbb{R}^d$ are the node embeddings of a_i, a_j as described above, and $A_k \in \mathbb{R}^{d \times d}$ is the *reagent matrix*. Our task is the learn A_k for each reagent k in the dataset. This model corresponds to a chemical intuition that each reagent corresponds to a *linear transformation* which maps source atom embeddings onto the direction of sink atom embeddings.

To exemplify the antisymmetric nature of bonding (i.e. a good source-sink pair is likely to be very poor as a sink-source pair), we also attempted to learn antisymmetric matrices for reagents (i.e. where $x^T Az = -z^T Ax$). However, this did not perform as well as a full matrix.

We implemented our logistic regression model using Tensorflow [8], optimizing a weighted log-loss using an Adam optimizer. In each iteration we performed gradient descent on the batch of examples corresponding to a single reaction.

4.2 Gaussian Discriminant Analysis

In our GDA, we assume that true bonding embedding pairs (x_i, x_j) are generated according to

$$x_i - x_j \sim \mathcal{N}(\mu_1, \Sigma_1)$$

and similarly for true nonbonding pairs. That is, we assume the *difference* between source and sink embeddings is distributed according to a multivariate normal, corresponding to the chemical heuristic that reactivity typically depends on the difference in electrical characteristics between two atoms, rather than their absolute values. For numerical stability on smaller datasets, we computed the MAP estimator for the covariance matrix Σ under a Wishart prior $\Sigma \sim \mathcal{W}(V = \frac{1}{d} I_d, d)$ in lieu of the MLE. For the estimates of μ, we used MLE. At prediction time, we predict $\arg\max_{(a_i, a_j)} p_1(x_i - x_j)/p_0(x_i - x_j)$ as the bonding pair, where $p_i(x) = \text{PDF}(X = x)$ where $X \sim \mathcal{N}(\mu_i, \Sigma_i)$.

4.3 Graphical Convolutional Network

In our graphical convolutional network model, we follow the same probabilistic approach in logistic regression, except that we learn the node embeddings using a graphical convolutional network from the node identity and context. [9] That is, we assume

$$P((a_i, a_j), R_k) = g(h_k(\hat{x}_i)^{\mathrm{T}} A_k h_k(\hat{x}_j))$$

where h_k is a graphical convolutional neural network corresponding to the reagent R_k and $\hat{x}_i$ is a simple embedding of atom a_i consisting only of a one-hot representation of its identity, aromaticity, and valence. We update based on the neighbor embeddings, the current self-embedding, as well as the bond types. For notational simplicity, let T be the set of bond types $\{\text{single, double, triple, aromatic}\}$ and let V_t be the set of edges corresponding to bonds of type $t \in T$. Then in layer n of the network, we update the embedding of a node x_i as

$$x_i^{[n]} := g\left(W_t^{[n]} x_i^{[n-1]} + \sum_{t \in T} \sum_{\{a_i, a_j\} \in V_t} W_t^{[n]} x_j^{[n-1]} \right)$$

where W_i is the *self-update* matrix that functions similarly to a memory gate in a RNN and g is once again the logistic function and is applied after every convolution except at the output. Our task is to learn the five Ws for each layer. We attempted to learn reagent-specific weight matrices W_k as well as a set of global weight matrices. However, the global approach did not perform as well as the reagent-specific approach. Furthermore, we vectorized using adjacency matrices and also attempted to normalize for degree by performing fast spectral decomposition on adjacency matrices, but this led to poorer results.

We implemented our graphical convolutional model using Tensorflow [8], using an Adam optimizer to optimize a weighted log-loss:

$$LL = 5y \log \hat{y} + (1 - y) \log(1 - \hat{y}).$$

In each iteration we performed gradient descent on the batch of examples corresponding to a single reaction. Also, due to overfitting concerns, we enforce a L2 loss term.

5 Results

A summary of best results achieved for each model on the testing set is shown in Table 1. We report *accuracy*, defined as the percentage of reactions for which the model correctly predicted the true bond as being the most likely pair. Note that this differs from the metric most papers in the field report, which is the percentage of reactions for which the true result was in the top N rated most likely by the model. This metric is called top-N accuracy, by which convention our metric is the top-1 accuracy.

5.1 Logistic Regression

We performed automated hyperparameter search on the following hyperparameters:

1. Matrix type. We examined full, skew-symmetric, and symmetric matrices, finding full matrices to yield the best results.
2. Class weight. We tested class weights of 5, 10, 20, and 40 and achieved best results with 5.
3. Learning rate. We tested learning rates of 0.001, 0.0005, 0.0001, and 0.00005 and found 0.0001 to achieve the best results.

Logistic regression achieved fast convergence (often < 30 epochs) on almost all reagents. Very little overfitting was observed and no regularization was necessary. Contrary to initial expectations, we found that the skew-symmetric matrices performed poorly. Despite the chemical validity of the underlying assumption, it is possible that fitting the skew-symmetric matrix involved a difficult non-convex learning problem due to the discontinuous formulation of the problem under a logistic output.

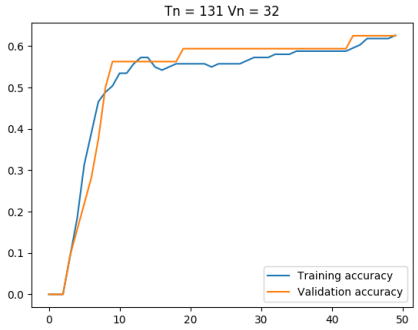


Figure 1: Sample logistic regression train/val accuracy on reagent HCl (03) over the course of training

5.2 Gaussian Discriminant Analysis

We visualized the training of the GDA model using t-SNE [10], a common dimensionality reduction technique that preserves distances between points, with the aim of observing clustering of the positive and negative classes. Such structure was apparent in some datasets (right) but not others (left). However, this is likely due to the nonreduceability of the distribution rather than a lack a structure in the underlying data, as no correlation between cluster visibility and classification accuracy was witnessed.

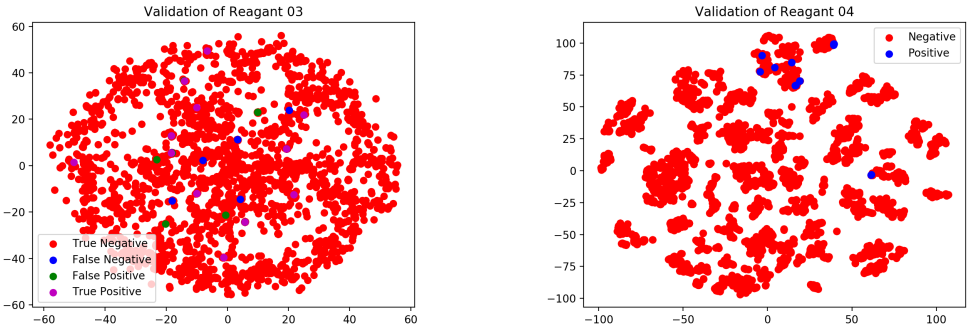


Figure 2: Plot of data for reagants Hydrochloric acid (03) and Triethylamine, carbonate (04) with dimensions reduced using the t-SNE technique.

5.3 Convolutional Graphical Network

We performed extensive automated hyperparameter search with the following hyperparameters:

1. Learning rate: We examined values in $\{1e{-}06, 2.5e{-}06, 5e{-}06, 1e{-}05, 2.5e{-}05, 5e{-}05, 1e{-}04\}$ and found that $5e{-}06$ gave the most consistent results.
2. Number of layers: We examined values in $\{3, 4, 6, 8, 10\}$ and found that 6 gave the strongest results. Higher numbers of layers tended to result in severe overfitting.
3. Regularization: We examined values in $\{0.00001, 0.001, 0.005, 0.01, 0.02\}$ and found that 0.005 gave the strongest results. Higher learning rates tended to result in lack of convergence.

Reagent	Name	Formula	Log. Reg	GDA	GNN
00	(no reagent)	(n/a)	48.5%	16.4%	**65.3%**
01	Chloroform	CCl_2H_2	42.7%	15.7%	**62.9%**
02	*N,N*-dimethylformamide	$(CH_3)_2NC(O)H$	**60.0%**	22.1%	59.7%
03	Hydrochloric acid	HCl	59.7%	18.1%	**69.4%**
04	Triethylamine, carbonate	$C_6H_{15}N, CO_3^-$	**83.2%**	30.4%	78.2%
05	*N,N*-dimethylformamide, carbonate	C_3H_7NO, CO_3^-	48.9%	13.4%	**74.6%**
06	*N,N*-dimethylformamide, carbonate, sodium hydride	$C_3H_7NO, CO_3 -, NaH$	63.4%	9.8%	**74.5%**
07	*N,N*-dimethylformamide, carbonate, water	$C_3H_7NO, CO_3 -, H_2O$	55.7%	22.4%	**69.3%**
08	Carbonate, Water	H_2O, CO_3^-	**60.0%**	31.9%	46.8%
09	Bicarbonate	HCO_3^-	40.9%	17.0%	**42.6%**
Weighted Avg			**52.2%**	**17.9%**	**65.2%**

Table 1: Model Accuracy

6 Discussion and Conclusion

The GNN generally performed better than the naive-embedding logistic regression, which in turn performed better than the naive-embedding GDA. However, logistic regression occasionally performed better than the GNN, corresponding to reagents for which the manually selected naive embedding was well-suited.

In particular, we find logistic regression performs well on reagents that are strongly acidic or basic (e.g. dibutylamine, triethylamine, or hydrochloric acid). This makes sense, as such reagents generally have simple behavior that depends on one or two major functional groups in each molecule, which is superficially encoded by the naive embedding. However, performance is worse on reagents that have no specifically predictable

function of their own, but serve as catalysts to bring out some inherent nucleophilic, acidic, or basic character in a molecule (e.g. the weak base carbonate). Such complex functions are typically due to nuanced structural characteristics (e.g. aromaticity, stereochemistry) that are lost by the simple embedding.

Therefore, due to this hardcoded manual embedding which is well-suited for some reagents but not for others, the logistic regression's performance was highly heterogeneous. In contrast, the GNN learns the most appropriate embeddings for each reagent and therefore achieves more homogeneous performance, but due to its slower learning rate in some cases it did not converge to a superior embedding within a reasonable number of epochs. The GDA uniformly performed poorly, likely due to the distribution of embedding differences not being modelled well by a Gaussian.

In future work, we would like to leverage the power of node embeddings to perform transfer learning across reagents. That is, using weights learned on a set of reagents, it may be possible to more quickly learn the feature embeddings relevant for a new reagents, given a large enough output dimensionality. We would also like to implement the message passing model of Battaglia et al [11] in order to endow *bonds* as well as molecules with properties which can be used in the embedding updates.

7 Contributions

Ethan Chi implemented the data-processing framework, the data scraper, and the GNN model.

Bowen Jing implemented the logistic regression model, as well as an automated hyperparameter search system.

Emily Wen implemented the GDA model and its associated analysis.

All authors contributed to the milestone, final paper, and poster.

8 Code

The repository for this project is available here: https://github.com/sabertooth-cat/chemistry.

References

[1] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen, "Prediction of organic reaction outcomes using machine learning," *ACS central science*, vol. 3, no. 5, pp. 434–443, 2017.

[2] J. Law, Z. Zsoldos, A. Simon, D. Reid, Y. Liu, S. Y. Khew, A. P. Johnson, S. Major, R. A. Wade, and H. Y. Ando, "Route designer: a retrosynthetic analysis tool utilizing automated retrosynthetic rule generation," *Journal of chemical information and modeling*, vol. 49, no. 3, pp. 593–602, 2009.

[3] W. Jin, C. Coley, R. Barzilay, and T. Jaakkola, "Predicting organic reaction outcomes with weisfeiler-lehman network," in *Advances in Neural Information Processing Systems*, 2017, pp. 2607–2616.

[4] D. Fooshee, A. Mood, E. Gutman, M. Tavakoli, G. Urban, F. Liu, N. Huynh, D. Van Vranken, and P. Baldi, "Deep learning for chemical reaction prediction," *Molecular Systems Design Engineering*, 2018.

[5] M. Segler and M. Waller, "Neural-symbolic machine learning for retrosynthesis and reaction prediction," *Chemistry 8211*, 2017.

[6] P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. Bekas *et al.*, "Molecular transformer–a model for uncertainty-calibrated chemical reaction prediction," 2019.

[7] D. M. Lowe, "Patent reaction extraction," 2014. [Online]. Available: https://bitbucket.org/dan2097/patent-reaction-extraction/downloads

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[10] L. van der Maaten, "t-sne." [Online]. Available: https://lvdmaaten.github.io/tsne/

[11] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.