Predicting Chemical Reactivity with Auto-Encoded Embeddings and Recurrent Neural Networks

Yan Leng, Quan Nguyen, and Lili Yu
Massachusetts Institute of Technology
Cambridge, MA
{yleng, qmn, liliyu}@mit.edu

***Abstract*—Knowing the steps and input reagents needed to synthesize a chemical is essential for industrial and research purposes. However, chemical reactions (and their papers) can be difficult to search. Furthermore, chemists cannot easily predict the reactivity of two chemicals if the database lacks the exact reaction. We pose this problem as a natural-language processing task on chemical names. First, we auto-encoded embeddings based on chemical names, relating similar words closer in the feature space. Then, we trained a recurrent neural network to predict whether two chemicals will react using a database with known chemical reactions (chemnet). We compared our model to a naive Bayes classifier, a maximum entropy / logistic regression classifier, and an SVM classifier trained on true reactions from the chemnet database interspersed with negative examples. Our model has many possible extensions, including the construction of novel synthesis pathways.**

I. Introduction

A. Motivation

Problems in chemistry are very complex – they consist of intricate chemical processes involving numerous molecules with various properties that can only be explored with many experiments. Chemometrics, which links statistical methods and chemical problems, has been a valuable tool in navigating chemical data and solving complex chemical problems. For example, partial least squares and multiple linear regression have been used in some chemical studies [1]. With their recent resurgence in other fields, artificial neural networks have seen increasing popularity in chemistry. Artificial neural networks can model the nonlinear empirical relationships commonly found in chemistry. However, even though many important problems in chemistry have been addressed, it remains an area where there is great room for improvement.

B. Previous Work

In 1990, [2] used neural networks to predict "the products of electrophilic aromatic substitution." The applications of neural networks in chemistry declined through the late 1990s, perhaps due to the lack of hardware resources capable of extensive training and the lack of data; the paper trains and tests on only 45 reactions. [3] in 1991 expressed "mixed feelings" about neural networks, concerning themselves with structural details like activation functions and biases. Both of these works predated the conception of long short-term memories (LSTMs), which greatly relieved the vanishing gradient problem. This enables us to build large neural networks that can extract something meaningful from the training data, even if important information is found at different places in the sentence.

In [4], neural grammar networks predicted chemical reactivity based on the SMILES and InChi descriptors of molecules. In [5], neural networks were used on 2D chemical fingerprints to predict "biological activities of structurally diverse chemical ligands." These fingerprints, which are bit vectors between 166 and 1024 bits in size, can be automatically generated or assembled from features identified by hand. However, constructing features by hand at the atom level, as done by [6], may fail to reveal properties conferred by the molecular macrostructure. Furthermore, no method yet has predicted general chemical reactivity, much less with the application of natural language processing (NLP) techniques, with neural networks.

C. Contribution

We propose a novel mechanism for predicting the reactivity of organic compounds. Rather than painstakingly

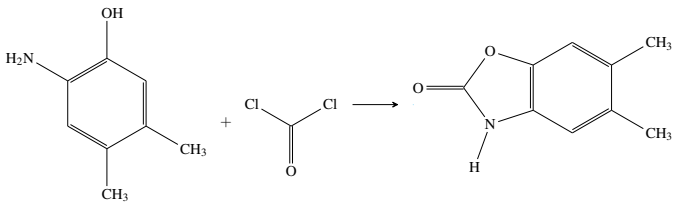

Fig. 1. An organic reaction found in the chemnet database. The reaction depicted here comes from a 1949 paper in which 2-amino-4,5-dimethylphenol and phosgene react to form 5,6-dimethyl-3h-benzooxazol-2-one [16].

collect chemical properties and hand-made features or construct grammars, we use a simpler approach: treat chemical names as sentences, and apply NLP techniques. We generate embeddings of chemical names in a high-dimensional space (akin to word2vec [7]) and use a recurrent neural network that employs these embeddings to determine whether two chemicals react.

II. Methodology

A. Problem Formulation

We form our chemical reaction binary classification problem as a language model with an exact mapping between letters to chemical elements, words to components of chemical names, phrases to chemical names, and reactions to sentences. The validity of chemical reaction is the same as whether two chemicals can react or not.

B. Data Preparation and Preprocessing

We obtained a network of organic chemicals and their reactions from a database called chemnet. Our subset of this network contains over 8,000,000 chemicals and 1,000,000 reactions with varying numbers of input reagents and output products. We limited the scope of our project to binary reactions: those with exactly two inputs and one output.

To generate training and test data, we selected 50,000 reactions from the database and collected all the elements used in these reactions. From this list of chemicals, we generated random pairs of reagents to form 50,000 false reactions after ensuring that the randomly selected chemicals did not participate in a known reaction in the database.

C. Embeddings

Before performing any training, we tokenized the chemical names, which can be as complex as "seq-trans-3-ethoxycarbonyl-2-isopropyl-2-methyl-oxiran". We split tokens based on the presence of not only whitespace but also hyphens. We also parsed parentheses and square brackets, which denote structural information about the chemical [8].

We treat chemical names as a language problem. They have very similar hierarchical structure. Building from the bottom up, chemical elements correspond to letters, chemical names to words, and chemical reactions to full sentences. Thus, we can build a model for chemical reactions using techniques from NLP.

The first step in building a language model is to find an effective way to represent words. The representation of a word, the basic unit of language, and essential for understanding more complex sentences, is critical in NLP. The vocabulary from a set of documents are, in general, very large. In many rule-based NP models, each word is represented as a one-hot vector (a bag-of-words (BOW) model). In this model, the relationship between different words and information about their ordering in the sentence are absent. Recently, probabilistic models, in which a word is predicted from its context, have shown great success in achieving a more meaningful representation and improves performance on various NLP problems. In this case, a word is represented as a low-dimensional real-valued dense vector. Each dimension of these representations captures latent information about a combination of syntactic and semantic word properties. Thus, we want to represent chemical names in a similar fashion.

However, just as there are arbitrarily numerous valid sentences with any number of words, there are arbitrarily many combinations of organic chemicals with any number of chemicals [9]. Thus, it is hard to tell whether two selected chemicals will react. This sparsity problem is analogous to the sparsity afflicting NLP's bigram and trigram models.

D. Recurrent Neural Networks

Recurrent neural networks (RNN) take in input sequentially with no assumption on the . Since an RNN reads data sequentially, the context of the chemical elements will be retained. Traditional machine learning, based on bags of words, suffers from the loss of word ordering in a sentence. As a result, running a traditional NLP algorithm on a chemical name would lose the information about chemical structure.

1) RNN Structure: For our project, we developed two RNN architectures. The first one takes word embeddings generated from an auto-encoder from previous step as a fixed word vector. The other takes one hot vectors as input, with three layers including: one embedding layer, one LSTM layer and one output layer. We trained combinations of chemicals on the whether these two chemicals will react. The following graphs show the structure of the two RNNs.

As stated before, RNNs can take in information sequentially and handle long-term dependencies. However, conventional RNNs suffer from the vanishing gradient problem, especially over long sequences of input. Fortunately, the use of Long Short-Term Memories (LSTMs) can help.

E. Implementation

The baseline models (naïve Bayes, logistic regression, and SVM) were created with scikit-learn [10], a machine learning toolkit for Python. The recurrent neural network models were built with Keras [11], a neural network library built on Theano [12] [13].

In the first model, depicted in Figure 3, we created an auto-encoder to derive 100-dimensional dense vector representations of chemical words. Words are passed (as indices into a vocabulary) through an embedding layer, an LSTM layer, and an output layer to a softmax distribution over the possible vocabulary (roughly 23,000 words in size). We trained the auto-encoder for 100 iterations on 100,000 chemical names derived from the same database (although not necessarily with the chemical names used in the 100,000 chemical reactions mentioned earlier). Then, we used these auto-encoded embeddings in a second RNN to perform the binary classification.

In the second model, depicted in Figure 5, we embedded both chemicals directly and passed them through separate LSTM layers. We used a "Siamese" neural network architecture [14], which ties the embedding layer and LSTM layer weights together, to ensure consistent embeddings. Then, we concatenated the output of these two LSTMs and fed them through a third LSTM before performing the binary classification.

Most training was performed on a GeForce TITAN X GPU, with use generously provided by the NLP group in CSAIL. Performance on the training set, when we allowed the model to adjust its own embeddings, leveled off at about 91%. The development set accuracy leveled off at about 85%. As shown in Figure 6, performance on the model with fixed embeddings fared no better than the naïve Bayes classifier.

III. Evaluation

Our results are summarized in Figure 7.

A. Understanding the Embeddings

Tokenization of the chemical names exposed similarities between chemical name structure and human languages. Despite conforming to a nomenclature specified by IUPAC, tokens of chemical names (in the reactions named in the training set) follow Zipf's Law: the word "acid" appears over 30,000 times, while the next most common word, "ester", appears 15,000 times. This uncanny adherence to Zipf's Law affirms our belief that techniques applicable to natural language processing can also be used for analyzing chemical names.

However, there are a few differences. In natural language, the most frequent words tend to be meaningless prepositions or determiners such as "the," "of," "and," and "a." In this "chemical" language, words reflect core structure, such as "acid," "ester," "ethyl" or common functional groups such as "chloride" and "bromide." This gives us two important pieces of information. First, it is easier to model a chemical problem than a conventional language problem, since most information is carried by common chemical subgroups. Second, the language of chemical names better reflects a hierarchical structure because it starts with a core structure which is then modified by functional groups. To draw comparisons to computer vision, the core structure of a chemical is like edges with different orientations in a image (which carry the most abstract information), while functional groups are like segments in a image (which carry more concrete information).

B. Regression over Chemical Fingerprints

We use chemical fingerprints to evaluate the performance of the information from our word embeddings. Chemical fingerprints are human-generated bit vectors that encode various chemical substructures. These fingerprints are implemented by several different databases, mainly for the purpose of neighboring and similarity searching. For example, a PubChem CACTVS chemical fingerprint is "an ordered list of binary (1/0) bits. Each bit represents a Boolean determination of, or test for, the presence of, for example, an element count, a type of ring system, atom pairing, atom environment (nearest neighbors), etc., in a chemical structure" [15]. [6] demonstrated the usefulness of chemical fingerprints in drug discovery. We performed regression on chemical fingerprints from PubChem to see how well our automatically-learned embeddings could correspond to chemical features. Figure 9 shows the accuracy of using word embeddings to predict fingerprist. It has overall accuracy of 0.91 over 900 dimensions and we found it has lower performance from dimension 320 to 720, which correspond to an atom's nearest neighbors. These features only naïvely detect atom pairs and we believe these are not good features and it is reasonable that word embeddings would not perform well. The average accuracy from dimension 320 to 720 is 0.87 while the average over all other dimensions is 0.96. This evidences that the word embeddings can effectively represent chemical structure properties.

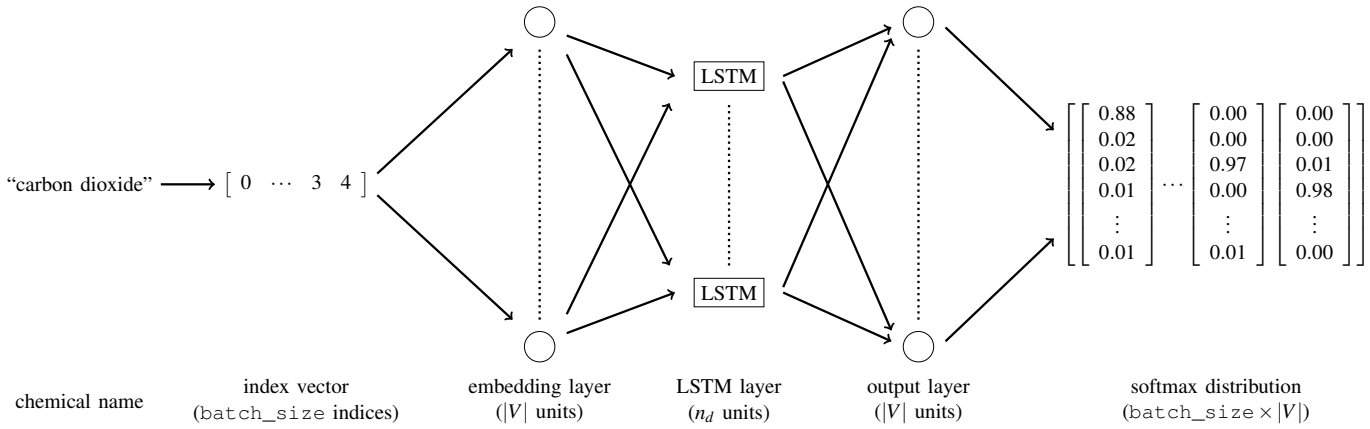

Fig. 2. Architecture of the neural network auto-encoder.

One minor problem in using fingerprints is that fingerprints can be implemented differently by databases. Many portions of fingerprints overlap with each other, and the large number of fingerprints (usually around 1000) makes searching and computation inefficient.

C. Clustering Chemical Names

To qualitatively assess the performance of our model, we clustered a few chemical names and examined their molecular structure. We hypothesize that chemicals with similar properties will occur close to each other in the vector space. For example, in Figure 10, several substructures have a 5- or 6-carbon ring, and a nitrogen-hydrogen (an "N-H" bond).

D. Baseline Model Performance

Our baseline models include a naïve Bayes, logistic regression, and support vector machine (SVM) classifiers. The features use the chemical names as bags of words.

1) Naïve Bayes: The naïve Bayes classifier is based on the assumption that all words, in this case, portions of a chemical name, occur independently from each other. However, this is not always true: the hydroxide ion (OH^-) occurs frequently in practice, so the presence of an oxygen atom would increase the probability of the appearance of a neighboring hydrogen atom.

However, given the nice performance of naïve Bayes (the same as the RNN with pre-trained word embed-

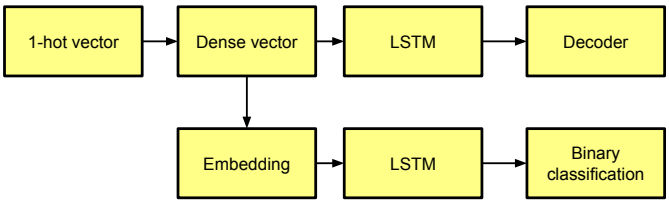

Fig. 3. The lower RNN uses the embeddings generated by the auto-encoder. The detailed architecture of the auto-encoder is shown in Figure 2.

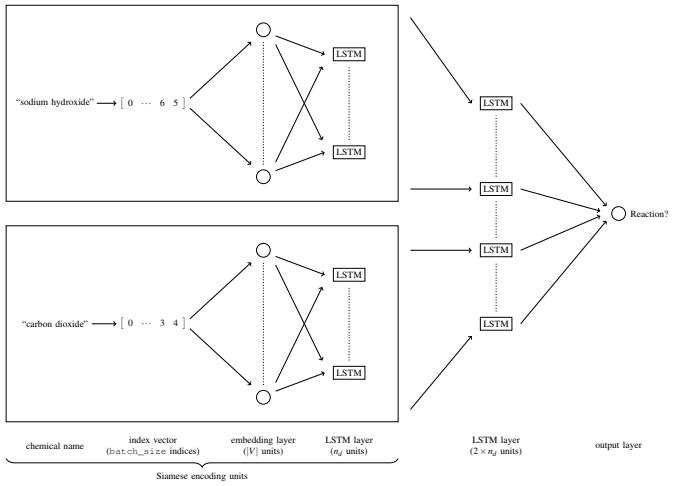

Fig. 4. Architecture of the neural network that trains its own embeddings.

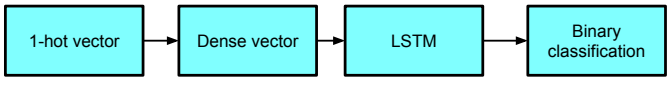

Fig. 5. This RNN trains the embeddings against the supervisory signal of whether two chemicals react. The detailed architecture is shown in Figure 4.

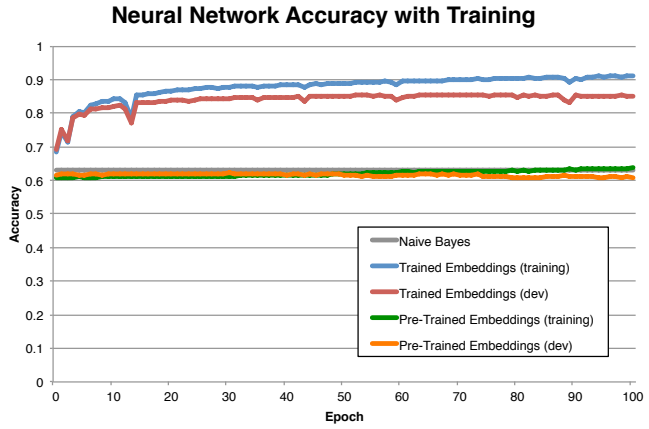

Fig. 6. Accuracy of the neural network models increases with each epoch.

Model	Training	Test
Random Guess	0.500	0.500
Naive Bayes	0.635	0.628
Maximum Entropy	0.635	0.601
Support Vector Machine	0.500	0.498
RNN with Fixed Embeddings	0.636	0.608
RNN with Trained Embeddings	0.912	0.851

Fig. 7. Binary classification scores. Values identify accuracy in determining whether a reaction will occur.

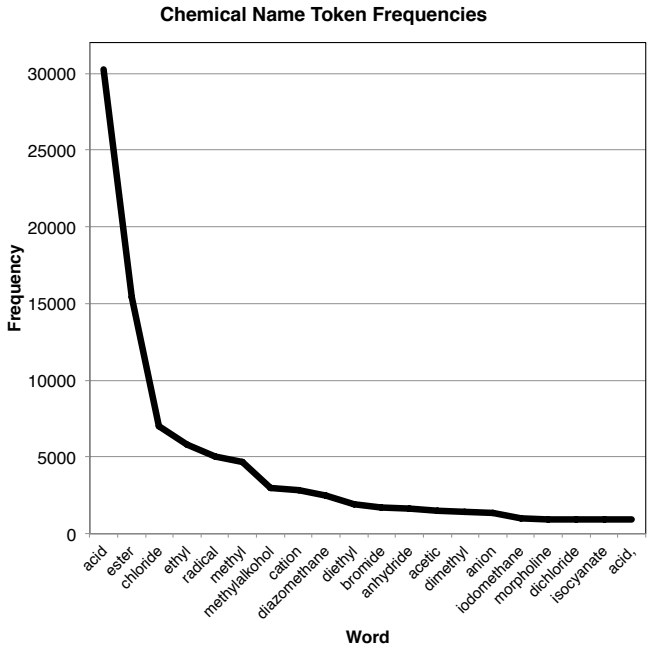

Fig. 8. Tokens in chemical names follow Zipf's Law.

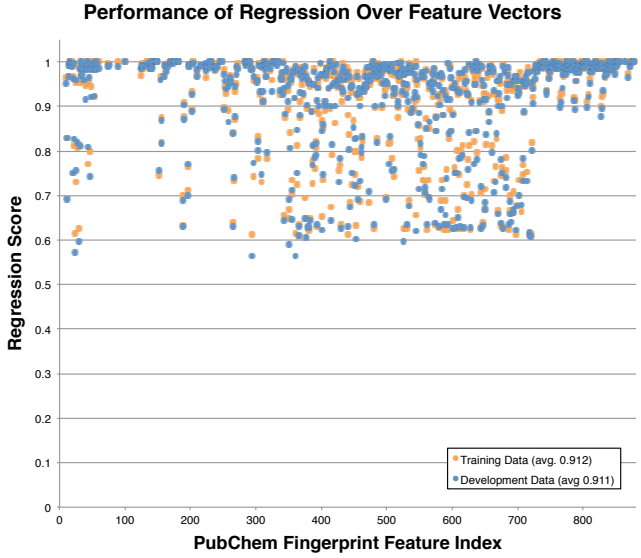

Fig. 9. Scatter plot of the effectiveness of our embeddings to fit to the features in the PubChem CACTVS Fingerprint.

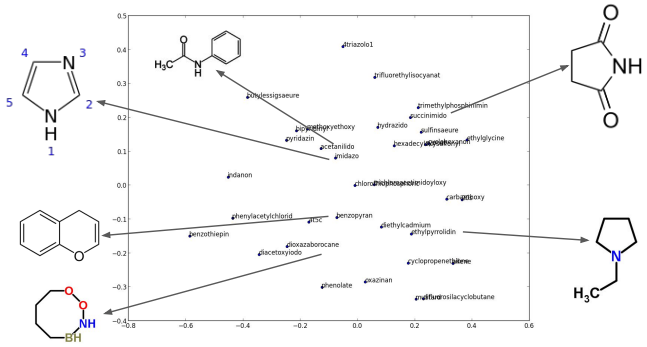

Fig. 10. Words from a cluster in the 100-dimensional space, reduced to two dimensions with PCA, and plotted.

dings), we can hypothesize that the classifier knows something about the chemicals that react and those that do not. It could be possibly predicating its classification on the presence of a particular sequence of chemical tokens.

2) Logistic Regression: Logistic regression performs as well as naïve Bayes with slightly higher performance on the training set, but lower performance on the development set.

3) Support Vector Machine: We implemented SVM with linear and RBF kernels. However, it suffers both from overfitting and high computational cost. With poor performance on the test set, we think SVM may not serve well as a machine learning technique in the domain of

chemistry.

E. RNN Model Performance

The accuracy of RNN increases sharply in the first 8 epochs and continues to increase slowly after the tenth epoch. We stopped at epoch 100, but we expect performance to increase with more training data and more training epochs.

F. Parameter Tuning

We tuned learning rates, dropout rates, and the dimension of the word embeddings. We tuned our parameters on a smaller set of 10,000 samples with 10 epochs, to reduce testing time. As expected, the accuracy is much lower than what we get using 100,000 observations with more epochs.

1) Learning Rate: We tuned the learning rate to the values [0.001, 0.01, 0.1, 1] in Figure 11. We found that accuracy on the test set is highest with the learning rate of 0.1. The loss dropped sharply at the second epoch. The losses for the four epochs do not vary much.

2) Dropout Rate: Dropout rate serves to regularize parameters and effectively reduces overfitting. We test dropout rates from 0.5 to 0.9 in Figure 12. We found that the higher the dropout rates, the better the performance, meaning that low dropout rates cause overfitting, a common but serious problem in complex RNN architectures. For the loss in each epoch, we found that the higher the dropout rate, the quicker it drops after the first epoch.

3) Word Embedding Dimensions: We tune the size of word embeddings in Figure 13. The accuracy on training set does not vary too much from the test set. However, the performance of test set decreases when we increase the number of word embedding dimensions. The best accuracy is achieved when the size of embedding is 50. The accuracy for 100, 200, 400 and 600 dimensions are similar.

4) Hidden Layer Dimensions: The accuracy, as depicted in Figure 14 reaches the highest value when the hidden layer has 100 hidden nodes. The accuracy decreases after we have more hidden nodes, indicating an overfit. By looking at the losses, we can see that the loss is relatively stable.

IV. Conclusion

Predicting chemical reactivity appears to be well-posed as a natural language processing problem, which can not only be validated by the performance of our models but also by our domain's adherence to Zipf's

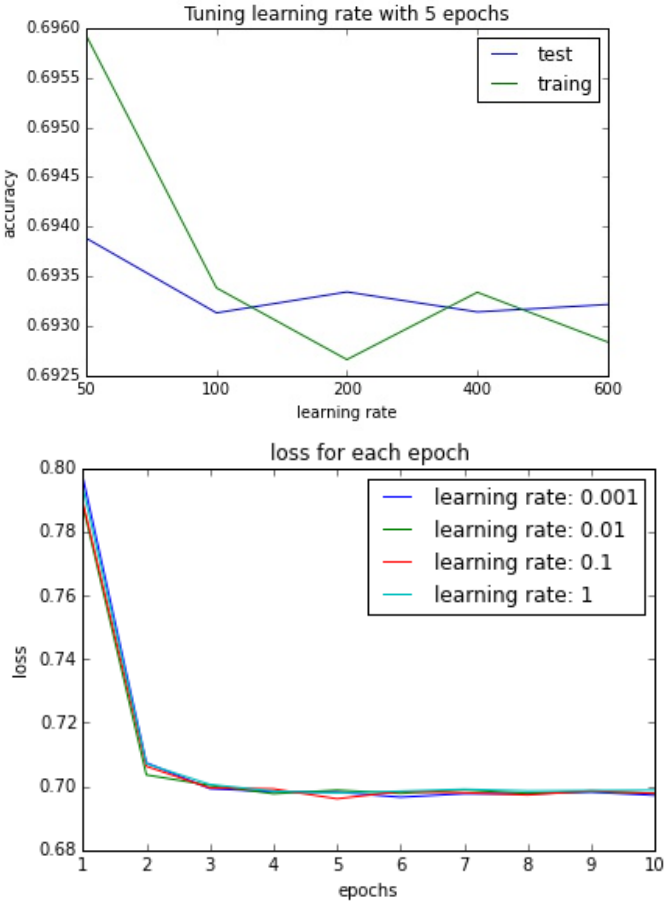

Fig. 11. Accuracy and training loss as a function of learning rate.

Law. Our exploration proved the possibility of learning new domain knowledge in a specific area, like chemistry, by treating it as a natural language problem.

Some conclusions from the observations of the results include:

- From word embedding we can see that similar chemicals are close in the vector space learned from the chemical names.
- Supervised training of word embeddings generate meaningful word vectors by improving 77.2% over the traditional machine learning methods.
- An RNN with pre-trained embeddings from chemical names performs as well as a naïve Bayes classifier. The surprisingly good performance of naïve Bayes may indicate that some reactions may be predictable by the presence of combinations of elements.

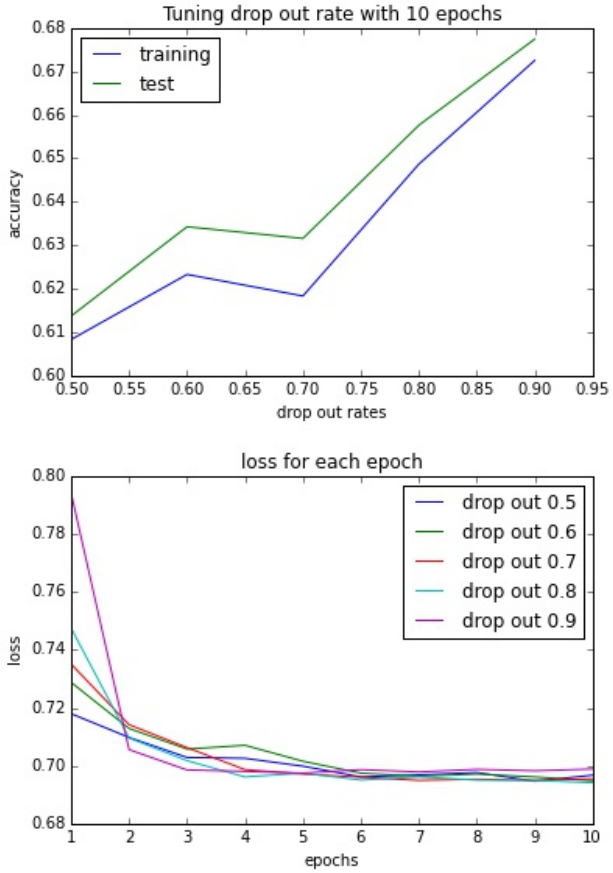

Fig. 12. Accuracy and training loss as a function of dropout rate.

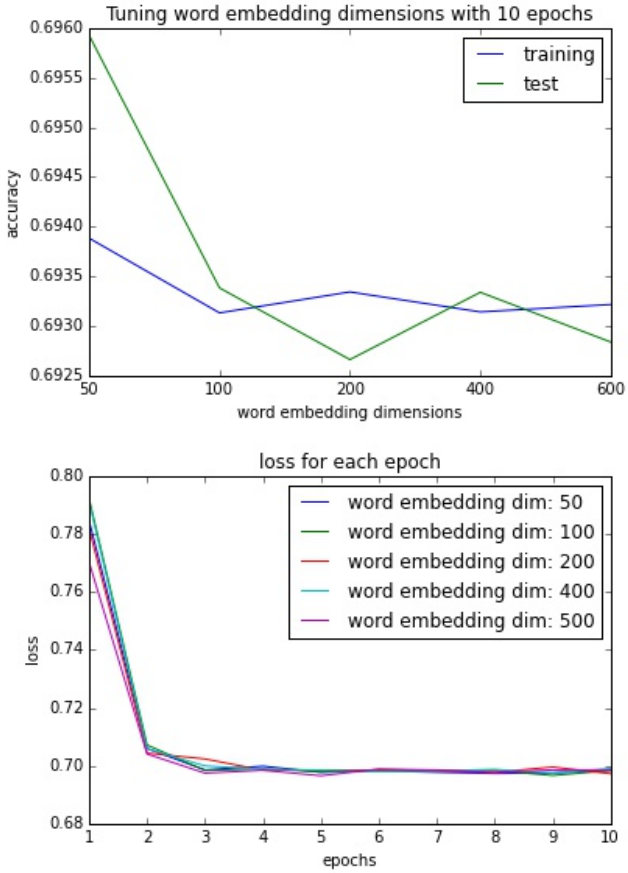

Fig. 13. Accuracy and training loss as a function of embedding vector dimensions.

A. *Future Work*

Due of limited time and access to data, we failed to implement lots of our ideas to improve our model:

- Training over bag of words: The core part of a modeling problem is the model itself and representation of the data. We are very confident with the effectiveness of the LSTM structure because of its improved performance over baseline models. However, it would interesting to measure the performance and speed benefits by moving from a one-hot vector to a dense vector representation.
- Hyperparameter tuning: We explored the hyperparameters of our models such as the dropout rate, word embedding dimensions, learning rate, and the number of hidden units. However we only compared the performance over the first several epochs, which may not predict the long-term effects of model training.
- Architecture selection: One shortcoming of the RNN is its sequential processing of data. The output up to a time step only contains information of elements before that time step. Bidirectional RNN are two separate RNNs stacked on top of each other: one receives input in the forward direction and the other receives input in the reverse direction. The output is computed based on the hidden state of both RNNs. We want to explore bidirectional LSTMs, which show improved performance for tasks like speech recognition.
- Data visualization: It is important to understand what is going on in the neural network by visualizing the data and the model. An auto-decoder can change a dense vector to a one-hot vector in the vocabulary. By decoding our dense vectors, we can understand the meaning of the LSTM outputs. This may also let us compare the output of the LSTM and the final product in real reaction. However, fine-tuning such a mechanism is problematic because it requires far more training than the binary classification used in this project.

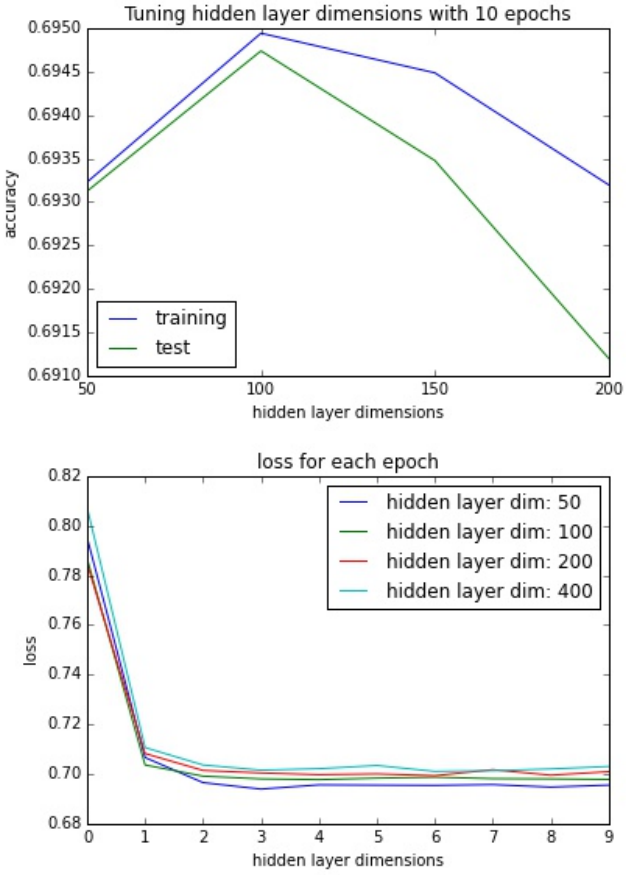

Fig. 14. Accuracy and training loss as a function of hidden layer dimensions.

In the future, we can see more chemical problems being explored as an "unknown" language with deep learning and artificial neural networks. One we understand chemical reactivity, we can predict chemical outputs. With the outputs of chemical reactions, we can assemble plausible sequences of chemical combinations to develop novel synthesis pathways for all sorts of chemicals. Learning new ways to synthesize chemicals can revolutionize not only the field chemistry, but also medicine and commerce.

REFERENCES

[1] Almeida, M. O., et al. "Medicinal electrochemistry: Integration of electrochemistry, medicinal chemistry and computational chemistry." *Current medicinal chemistry* 21.20 (2014): 2266-2275.

[2] Elrod, David W., Gerald M. Maggiora, and Robert G. Trenary. "Applications of neural networks in chemistry. 1. Prediction of electrophilic aromatic substitution reactions." *Journal of chemical information and computer sciences* 30.4 (1990): 477-484.

[3] Zupan, Jure, and Johann Gasteiger. "Neural networks: A new method for solving chemical problems or just a passing phase?." *Analytica Chimica Acta* 248.1 (1991): 1-30.

[4] Ma, E.Y.T.; Kremer, S.C., "Neural Grammar Networks in QSAR Chemistry," in *Bioinformatics and Biomedicine, 2009. BIBM '09. IEEE International Conference on* , vol., no., pp.37-42, 1-4 Nov. 2009 doi: 10.1109/BIBM.2009.60

[5] Myint, Kyaw-Zeyar et al. "Molecular Fingerprint-Based Artificial Neural Networks QSAR for Ligand Biological Activity Predictions." *Molecular pharmaceutics* 9.10 (2012): 29122923. PMC. Web. 12 Dec. 2015.

[6] Durant, J. L.; Leland, B. A.; Henry, D. R.; Nourse, J. G. Reoptimization of MDL Keys for Use in Drug Discovery. *J. Chem. Inf. Comput. Sci.* 2002, 42, 12731280.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*, 2013.

[8] Nomenclature of Organic Chemistry, Sections A, B, C, D, E, F, and H, Pergamon Press, Oxford, 1979. Copyright 1979 IUPAC.

[9] Wikipedia contributors. "Organic chemistry." Wikipedia, The Free Encyclopedia. *Wikipedia, The Free Encyclopedia*, 22 Nov. 2015. Web. 12 Dec. 2015.

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] http://keras.io/

[12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. "Theano: new features and speed improvements". NIPS 2012 deep learning workshop.

[13] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. "Theano: A CPU and GPU Math Expression Compiler". *Proceedings of the Python for Scientific Computing Conference (SciPy)* 2010. June 30 - July 3, Austin, TX.

[14] Bromley, Jane, et al. "Signature verification using a 'Siamese' time delay neural network." *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993): 669-688.

[15] ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.txt

[16] Close, W. J., Burris D. Tiffany, and M. A. Spielman. "The analgesic activity of some benzoxazolone derivatives." *Journal of the American Chemical Society* 71.4 (1949): 1265-1268.