Deep Learning Approaches to Chemical Property Prediction from Brewing Recipes

Gracie Ermi,[*][†] Ellyn Ayton,[‡] Nolan Price[‡] and Brian Hutchinson[‡][§]

[†]Vulcan Inc., Seattle, Washington
[‡]Western Washington University, Bellingham, Washington
[§]Pacific Northwest National Laboratory, Richland, Washington

***Abstract*—Despite the explosion of craft beer brewing over the last decade, there is virtually no work in the public domain exploring machine learning approaches to understand and optimize the brewing process. Learning to map between representations of an object across different domains is one of the fundamental challenges in machine learning. There are at least three distinct representations of beer that one may wish to learn to map between: 1) the brewing recipe, 2) the chemical composition of the resulting beer and 3) the written reviews of the beer. The mapping between any pair of these three domains is highly non-linear. Brewing beer involves complicated biological and chemical processes, while the human qualitative perception of a beer may be even more complex. In the work described in this paper, we focus on the former: mapping between the recipe and chemical attribute domains. We use two deep learning architectures to model the non-linear relationship between beer in these two domains, classifying coarse- and fine-grained beer type and predicting ranges for original gravity, final gravity, alcohol by volume, international bitterness units and color. Such models could be used to optimize recipes to produce desired chemical properties of the beer, allowing brewers to design better tasting beer, faster and with less waste. Using a set of approximately 223K brewing recipes from homebrewing site brewtoad.com, we find that deep and recurrent neural network models significantly outperform several baselines in predicting these attributes, offering relative reductions in classification error by 20%+ and reducing the root mean squared error for the attribute ranges by 44% relative to the best baseline.**

I. Introduction

Many important tasks require mapping mapping between distinct, highly non-linearly related domains; e.g., image classification, speech recognition. On task that has garnered relatively little attention is that of modeling the processes of beer brewing. Brewing too involves several distinct representations in non-linearly-related domains. In one view, a beer can be described by a recipe, which specifies the particular ingredients required, their quantities and the times at which they are added during the brewing process. In another view, a beer can be described by its objective chemical attributes; for example, color, bitterness, relative presence of various flavor and aroma compounds, etc. Finally, a beer can be viewed through the way it is perceived subjectively; e.g., in the words used to describe it in written reviews. This work focuses on one aspect of this domain mapping problem: being able to accurately predict the chemical attributes of beer from its recipe. Combined with a search through the recipe space, this mapping would allow rapid development of new beers, and allow brewers to optimize for flavor, brewing time and consumption of resources. There are many challenges, including the fact that recipes are of variable length, involve a large "vocabulary" of ingredients and are only partially ordered. We propose two deep learning approaches to this mapping: first, deep neural networks (DNNs) that take as input a bag-of-ingredients featurization of the recipe, and second, recurrent neural networks (RNNs) acting as encoders that consume ingredient sequences as input. For both models, we tackle three prediction tasks: a coarse-grained beer type prediction task (ale, lager or wheat), an 81-way fine-grained beer type prediction task (e.g. robust porter or american ipa), and a 10-dimensional regression task of important chemical properties of beer (e.g. measuring alcohol content, bitterness, or color). The next section provides some preliminary information on brewing.

[*]Work conducted as a student at Western Washington University.

II. Brewing Background

There are four major classes of ingredients in brewing recipes: fermentables, hops, yeasts and other miscellaneous additives. Fermentables (mostly malts) contain sugars that will be fermented by the yeast, producing alcohol. Lighter malts produce lighter-colored beer while darker malts (those that are toasted for a longer period of time) create a darker-colored beer. Hops, the flowers of the *humulus lupulus* plant, serve three major purposes: to add bitterness, flavoring and aroma. The yeast metabolizes sugars into alcohol; different strains of yeast produce distinctive flavor and aroma compounds. Any other ingredients added to the beer during the brewing process fall under the Miscellaneous category. These include special flavoring additives (e.g. vanilla, orange peel).

Brewing a batch of beer is a complicated process, sometimes taking months to complete. First, barley is milled (crushed) to expose the starchy contents of the barley seed. This milled grain is then added to a "mash tun," and combined with water at a specific temperature to activate enzymes that produce sugar. The mash is then filtered in a "lauter tun" to separate the sugary water, known as wort, from the grain. The *gravity* of the wort measures the amount of dissolved sugars in the water; the higher the gravity, the more sugar that is available to be converted into alcohol during fermentation. The wort is then brought to a sustained boil, usually for at least one hour. During the boil, different quantities and varieties of hops are added strategically. Hops contain alpha acids that are isomerized during the boil to produce bitterness. Different hop varieties have different amounts of alpha acids. Complicated factors affect the extent to which alpha acids are isomerized and dissolved into the boil, including the duration that the hops are boiled, the strength of the boil, and the wort's gravity. In practice, some hops are added early primarily for bittering, some are added toward the end of the boil to impart flavoring, and others may be added at the end of the boil or during fermentation to contribute both flavoring and aroma. The exact timing, quantities and varieties can produce vastly different-tasting beer. After boiling the wort (usually for one to two hours) any malt or hop particles are removed and the cooling process begins. It is crucial to cool the wort as quickly as possible to avoid forming undesirable flavor and aroma compounds. The wort is usually cooled to around 70° Fahrenheit for ales and to about 50° Fahrenheit for lagers.

After the wort has sufficiently cooled, the "original gravity" (OG) reading is taken, which will later be used to estimate alcohol by volume. Finally, the yeast is pitched into the cooled wort, possibly after oxygenating the wort to facilitate yeast growth. The yeast rapidly reproduces, metabolizing sugars and oxygen into alcohol and carbon dioxide, among other byproducts. This conversion process, referred to as fermentation, can range from taking several days to several months. At the end of fermentation, a "final gravity" (FG) reading is taken to assess how much of the sugar was converted by the yeast. At this point the beer may be filtered and then kegged, canned or bottled for distribution.

III. Prior Work

Limited prior work has been conducted applying machine learning to fermentation, including both beer and wine. Most of this work has been focused around recommendation and the analysis of reviews of various beverages. Lipton et al. [1] use RNNs to classify and generate beer reviews collected from review site BeerAdvocate[1]. They generate reviews character by character conditioned upon the author, specific beer, beer category, or sentiment. Carroll's thesis [2] introduces an application for the recommendation of wine close to a users location and based upon their previous indication of taste preferences. McAuley et al. [3] use a dataset of beer reviews from review site RateBeer[2] to analyze how a beer consumer's tastes will change over time. Among their findings are that lagers are more accessible and rated higher by novice beer drinkers while strong ales, such as India Pale Ales (IPAs), are rated very highly by experienced beer drinkers. Kiddon [4] worked in the general recipe space and built a Neural Sequence Generation Model (a neural language model) to take a final product and a list of ingredients and generate the recipe's text. Kiddon's model uses attention mechanisms to record which ingredients have already been used and which ingredients to reference next. Additionally, Kiddon et al. [5] use an unsupervised expectation-maximization approach to predict the next step in an instructional recipe. The recipe is divided into semantic groups and then one part of the recipe is fed into the model which predicts the next step. The next step is based on the last used verb, or if there was no last verb, the model introduces a new ingredient.

To our knowledge, no prior work has used machine learning to predict the chemical attributes of beer from brewing recipes. However, learning sequence representations has a long history. RNNs (e.g. Elman networks [6]) have been used for decades. In 1997, Hochreiter and Schmidhuber introduced Long Short-Term Memory (LSTM) networks [7], which have experienced a resurgence in popularity in the past decade owing to their ability to better model long sequences. Similar models (e.g. utilizing Gated Recurrent Units [8]) are also widely used. In quick succession, several researchers introduced RNNs to map variable length input sequences to fixed length vector representations [8]–[10]. Such networks are often referred to as "encoders" and paired with decoder networks that map the vector back into a variable length sequence; this strategy is particularly popular for machine translation.

IV. Approach

A. Problem

In this work, we focus on three prediction tasks that map from the domain of beer recipes to the chemical attributes of the resulting beer.

1) A three-way "coarse-grained" classification task. This task classifies a beer as one of three high-level categories for beer types: Ale, Wheat or Lager.
2) An 81-way "fine-grained" classification task. Here we learn to predict more specific type classification labels for beer recipes (e.g. American IPA, Dry Stout, Premium American lager, etc.). See Appendix for a complete list.
3) A multivariate regression task that predicts 10 style attributes of a beer given its recipe. These attributes are the minimum and maximum values for each of the following: original gravity (OG), final gravity (FG), alcohol by volume (ABV), International Bitterness Units (IBU), and color.

B. Featurization

We consider two feature representation strategies for the brewing recipes. The first approach represents a recipe as a bag of ingredients, producing a fixed-length vector suitable for most machine learning techniques. This representation is a straightforward, conventional way to represent a beer recipe while also capturing some relevant information about beer ingredients. Notably, this representation does *not* capture the

[1]https://www.beeradvocate.com/

[2]https://www.ratebeer.com/

sequential nature of a recipe, but it does express the unique collection of the individual ingredients that contribute to the resulting beer.

The second featurization strategy represents a recipe as a set of five ingredient sequences, one for each ingredient type (fermentable, boil hop, dry hop, yeast, miscellaneous), which can be consumed by five encoder RNNs. Each ingredient contains not only the type and quantity, but in the case of fermentables and boil hops, when it is added to the boil. Note that the brewing recipe is only partially ordered: several ingredients may be added at the same time. When two ingredients have no true order, we order them arbitrarily. Disregarding the sequential steps, ingredients could instead be modeled as sets (e.g. [11]); exploring that direction is left for future work. Due to the open vocabulary nature of recipes, sufficiently rare tokens are mapped to an out-of-vocabulary token, `<oov>`, for both approaches.

1) Bag of Ingredients: Our first featurization strategy represents recipes as a bag-of-ingredients. Each recipe vector is the concatenation of six subvectors:

1.1) A binary, multi-hot vector whose length is equal to the number of in-vocabulary yeast types. This indicates which yeasts are used in the recipe (often just a single yeast type). Note that we indicate only which yeasts are present and not their respective quantities, because the quantity of yeast is not particularly important.

1.2) A real-valued vector whose length is equal to the number of in-vocabulary fermentables. Each element indicates the quantity (in kg) of the fermentable.

1.3) A real-valued vector whose length is equal to the number of in-vocabulary hops. Each element indicates the quantity (in oz) of the hop added during the boil.

1.4) A real valued vector whose length is equal to previous vector, whose elements indicate time at which the hop is added to the boil (in minutes prior to end of boil).

1.5) A real-valued vector whose length is equal to the number of in-vocabulary hops. Each element indicates the quantity (in oz) of the hop used for dry-hopping. Time is not recorded for dry hops. Hops used in the boil can also be used as a dry hop in the same recipe.

1.6) A binary, multi-hot vector whose length is equal to the number of in-vocabulary miscellaneous ingredients. This indicates which miscellaneous ingredients are used in the recipe. Because there are not standard units for these ingredients, we do not include their quantity.

The concatenation of these subvectors yields a feature vector with length 3027. This number is the sum of the total number of different types for each type of ingredient (yeasts, miscellaneous ingredients, boil hops (x2), dry hops, and fermentables).

2) Recipes as Sequences: Our second recipe representation is as sequences of ingredients. Here we produce five sequences: one each for fermentables, dry hops, boil hops, yeasts and miscellaneous ingredients. Each boil hop is represented as the concatenation of a learned embedding vector for the hop type, a quantity (in kg) and time (in minutes) before the boil. Each dry hop and fermentable ingredient is represented as the concatenation of a learned embedding vector for the ingredient type and the quantity. Each yeast and miscellaneous ingredient is represented by a learned embedding vector for the ingredient type.

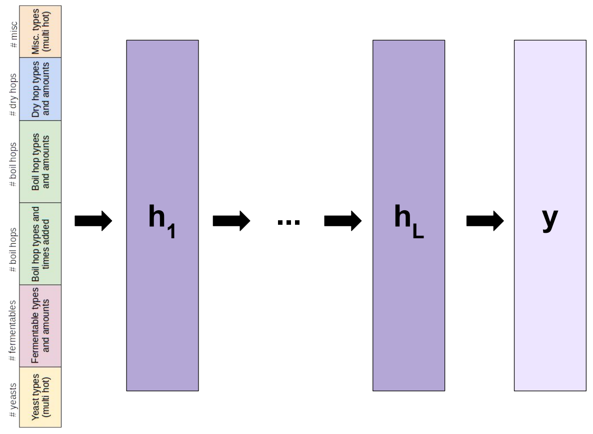

Fig. 1. The DNN model taking as input the bag-of-ingredients representation.

C. Deep Neural Network (DNN) Model

We first consider standard DNNs, which take as input the bag-of-ingredients feature vector, with L hidden layers:

$$h_{(\ell)} = g(W_{(\ell)}^T h_{(\ell-1)} + b_{(\ell)}) \text{ for } \ell = 1, 2, \ldots, L \tag{1}$$

where g is the hidden activation function (e.g. $\tanh$ or ReLU) and $h_{(0)} = x$ (the input). The trainable parameters are the weight matrices ($W_{(\ell)}$) and the biases ($b_{(\ell)}$). We use three DNN for our three prediction tasks: coarse-grained beer type classification, fine-grained beer type classification and beer attribute prediction (regression). For both classification techniques, our output layer, y is defined as

$$y = \text{softmax}(W_{(L+1)}^T h_L + b_{(L+1)}) \tag{2}$$

giving posterior a probability vector over the output classes. All model weights are trained with cross-entropy loss.

For the regression task, we use a linear (identity) output activation

$$y = W_{(L+1)}^T h_L + b_{(L+1)} \tag{3}$$

where $y \in \mathbb{R}^{10}$ gives the estimate of the 10 attributes we are predicting, and all model weights are trained to minimize the mean squared error loss. See Fig. 1 for an illustration of the DNN with the bag-of-ingredients features.

D. LSTM-DNN

The bag-of-ingredients representation loses the (partially) sequential nature of the recipe. Fortunately, RNN encoders

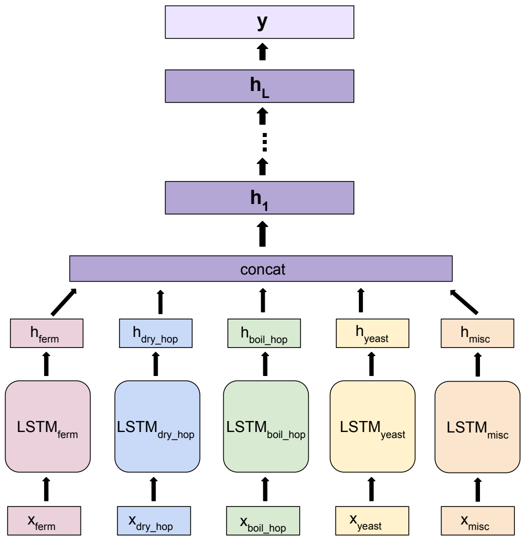

Fig. 2. The LSTM-DNN model. The boxes for the bottom three layers represent sequences; the last representation in each h sequence is concatenated to form "concat."

are quite effective at mapping variable length sequences to a fixed length representation. We use the popular LSTM model. Specifically, we train five separate LSTMs, one for each ingredient type. Each of the five LSTMs has the following standard form:

$$\begin{aligned}
i_t &= \sigma(W_{(i)}^T x_t + U_{(i)}^T h_{t-1} + b_{(i)}) && (4)\\
\tilde{c}_t &= \tanh(W_{(c)}^T x_t + U_{(c)}^T h_{t-1} + b_{(c)}) && (5)\\
f_t &= \sigma(W_{(f)}^T x_f + U_{(f)}^T h_{t-1} + b_{(f)}) && (6)\\
C_t &= i_t \circ \tilde{c}_t + f_t \circ C_{t-1} && (7)\\
o_t &= \sigma(W_{(o)}^T x_t + U_{(o)}^T h_{t-1} + V_{(o)}^T C_t + b_{(o)}) && (8)\\
h_t &= o_t \circ \tanh(C_t) && (9)
\end{aligned}$$

Here t is the time index and i, f and o denote the input, forget and output gates, respectively. The model parameters are the set of weight matrices (W, U, V) and the bias vectors (b). All weights are trained jointly. We concatenate the final h_t from each of the five LSTMs and feed that as input into a DNN. We refer to this combined model as the LSTM-DNN. Fig. 2 illustrates our multi-encoder LSTM-DNN model. Like our DNN model, the LSTM-DNN is used for two classification and one regression task, employing the same output activation and loss functions as the DNN.

We implemented both the DNN and LSTM-DNN using Google Tensorflow library [12], and train both using mini-batch stochastic gradient descent with the Adam optimization algorithm [13].

E. Baselines

We compare our DNN and LSTM-DNN against several baseline methods. All of our baseline models were trained using scikit-learn [14].

1) Classification Baselines: For the classification tasks, we consider three baseline methods. First, we use the majority class baseline, in which the model always predicts the class that was most common in the training set. Second, we use sckikit-learn's decision trees [15]. Third, we report results using multinomial logistic regression [16]. In addition to these baseline models, we considered Support Vector Machines [17], but they were prohibitively slow to train and thus not included in our comparison.

2) Regression Baselines: For the regression task, we again consider three baselines. The first baseline merely predicts $\bar{y}$, the average target (true output) over the training set. The second is a linear regression model trained to minimize the mean squared error loss. For our third regression baseline we use decision trees. As was the case with SVMs for classification, support vector regression [18] proved too slow to include in our study.

V. Experiments

A. Data

Our data consists of 228,596 XML formatted recipes taken from brewtoad.com. These recipes were created and published by homebrewers and many of the recipes were attempts to recreate, or clone, existing professional beers. Each recipe consists of the beer type, name, and a series of ingredients including yeasts, fermentable, hops, and miscellaneous ingredients. Additionally, the quantity and addition times for the ingredients were included. The recipes, after removing any duplicates, were randomly split 70%-10%-20% into train, development and test sets. All training set recipes were parsed and processed to produce an ingredient vocabulary, and then each recipe was encoded using one of the representations described in Sec. IV-B. The data consists of 87.9% ales, 5.6% lagers, and 7.0% wheat beers. The most common fine-grained type in the data is American IPA, making up 13.5% of the data. The 10 most popular fermentables, hops, yeasts and miscellaneous ingredients account for 14.7%, 35.2%, 46.1% and 39.1% of their ingredients categories, respectively.

B. Prediction Attributes

There are five chemical properties that we aim to predict in the regression task: the original and final gravities (OG and FG), the international bitterness units (IBU), the alcohol by volume (ABV), and the color. ABV itself is a function of OG and FG: $ABV = 133.62(OG - FG)$. IBUs are a standardized bitterness measurement: the higher the IBU, the more bitter the beer. Given known attributes of specific hop varieties, IBUs values can be estimated using a complicated, non-linear function known as the Tinseth formula. Color is measured on a scale known as the Standard Reference Method (SRM), which measures the attenuation of light of a particular wavelength. An SRM of 1 is the lightest (a pale yellow color), while and SRM of 40 is perceived as black. SRM values can be approximated using a non-linear Morey Equation. Our data does not contain the metadata required to estimate OG, ABV, IBUs or SRM; instead, our model must learn ingredient embeddings that encode the relevant information by inferring from recipes and attribute targets in the training data. All attributes were standardized prior to training using the training set mean and variance.

C. Metrics

We evaluate our style predictions by calculating the root mean squared error (RMSE) between our true values, $\hat{y}_{t_i}$, and our predictions, y_{t_i} averaged over all N data points:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|y_{t_i} - \hat{y}_{t_i}\|_2^2} \tag{10}$$

and evaluate the fine-grained and coarse-grained type predictions using both accuracy and perplexity. Perplexity (ppl) is cross-entropy exponentiated and is defined as follows

$$ppl = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{(i)k} \ln h(x_{(i)})_k\right) \tag{11}$$

where C is the number of classes, N is the number of datapoints in the set on which the evaluation is performed, $h(x_{(i)})_k$ is our model's estimate for the posterior probability of class k given the ith datapoint, and $y_{(i)k}$ is 1 if the true class for datapoint i is k and 0 for all other k. The lower the perplexity the better the model's performance. We report perplexities for the baselines when possible. For the "majority class" baselines, these models just output the training set's empirical distribution over the classes.

D. Tuning

We tuned our model hyper-parameters on the development set using grid search. For the DNNs, we tuned the hidden layer dimension, the learning rate, the range for randomly initializing our weight matrices, and the number of hidden layers. We fix the batch size to 32 samples for all three DNNs. For the coarse-grained classification, our best performance was achieved with a model whose weights were initialized within the range $[-0.1, 0.1]$, a learning rate of 0.1 and three hidden layers of size 100. Our best models for the fine-grained classification and style attribute regression tasks have similar hyper-parameter settings they use 200-dimensional hidden layers.

For the LSTM-DNN, we tune the learning rate, the weight initialization range and the number of DNN hidden layers. We again fix the batch size to 32.Our best performing models had identical hyper-parameter settings for all three tasks: weights were initialized within the range of $[-0.1, 0.1]$, the learning rate was 0.05, and two hidden layers of size 20 were used in the DNN component.

TABLE I
COARSE-GRAINED TYPE CLASSIFICATION RESULTS

	Accuracy	**Perplexity**
Majority Class	87.9%	1.58
Decision Tree	88.2%	-
Logistic Regression	88.9%	1.45
DNN	90.1% ± 0.1	1.35 ± 0.01
LSTM-DNN	**91.4%** ± 0.0	**1.26** ± 0.05

TABLE II
FINE-GRAINED TYPE CLASSIFICATION RESULTS

	Accuracy	**Perplexity**
Majority Class	14.4%	36.15
Decision Tree	27.6%	-
Logistic Regression	23.0%	21.06
DNN	27.6% ± 0.6	15.73 ± 0.54
LSTM-DNN	**34.3%** ± 0.7	**9.42** ± 0.24

E. Results

For all LSTM-DNN and DNN results, we report the mean and standard deviation of our performance metrics for five runs of the best set of hyper-parameters.

Table I summarizes model performance on the three-way coarse grained classification task. Performance is good for all models, owing to a large majority class, but is the highest for the deep learning models, with the LSTM-DNN giving a 29% relative reduction in error over the majority class baseline and a 23% relative reduction in error over the best baseline (logistic regression). The DNN and LSTM-DNN also yield lower perplexity, suggesting that they are producing better probability distributions over the output classes.

Table II shows the results for the 81-way fine-grained classification task. This is clearly a more challenging task, with the baselines averaging 22% accuracy. Still, the deep learning models give substantially better performance, with the DNN obtaining 28% accuracy and the LSTM-DNN giving 34%. The LSTM-DNN's perplexity is 74% lower than the perplexity given by the training set's empirical distribution ("majority class" perplexity).

Table III shows the results for the chemical property regression task, breaking down RMSE per attribute. While all of the baselines had RMSEs greater than one standard deviation, all of the deep models had RMSEs less than it. Again, the LSTM-DNN gave the best performance, with a 44% relative reduction in overall RMSE over the best baseline. Separating out the RMSE by individual attributes reveals where the majority of our error is concentrated. The max and min values for original gravity appear to be the most difficult of the 10 attributes to predict, followed by the minimum final gravity value.

To better understand what the models learn, Fig. 3 visualizes the learned recipe embeddings with the LSTM-DNN on the fine-grained type classification task. Each point is a recipe in the development set; its location is given by the final hidden representation projected down to two dimensions

TABLE III
RMSE FOR ATTRIBUTES PREDICTION (REGRESSION) (USING STANDARDIZED ATTRIBUTES)

	OG		**FG**		**ABV**		**IBU**		**Color**		**Overall**
	min	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	
Average	0.99	1.09	1.07	1.07	0.99	1.04	1.01	1.09	1.05	1.13	1.11
Lin. Regression	1.15	2.08	1.04	1.51	1.15	2.16	1.44	2.23	1.20	1.85	2.69
Decision Tree	1.19	1.19	1.16	1.20	1.18	1.18	1.10	1.14	0.92	1.02	1.28
DNN	0.93	0.91	0.92	0.93	0.93	0.91	0.86	0.88	0.85	0.86	0.90 ± 0.002
LSTM-DNN	**0.91**	**0.83**	**0.70**	**0.52**	**0.58**	**0.45**	**0.46**	**0.47**	**0.48**	**0.49**	$\mathbf{0.62 \pm 0.02}$

using T-SNE [19]. To reduce clutter, only the recipes from the top 11 most common types are plotted. Despite the dimensionality reduction, it shows intuitive trends: american IPAs and imperial IPAs are heavily overlapping; the various porters are nearby to each other, and saisons are the biggest "outlier" class. Similar clustering occurs when plotting the coarse types using the recipe embeddings learned by the coarse-grained classification LSTM-DNN.

VI. CONCLUSIONS

In this work, we present two deep learning approaches to predicting chemical attributes of beer from its brewing recipe: the first, a deep neural network applied to a bag-of-ingredients representation, and the second, an LSTM-DNN with separate LSTMs to encode ingredient sequences. Both models are evaluated on three task: classifying coarse-grained beer type, classifying fine-grained beer type, and predicting 10 real-valued beer attributes. On all three tasks, the deep learning models outperformed standard baselines, with the LSTM-DNN model giving the best performance. By visualizing the hidden representations, we observe that the model is learning meaningful structure in the space of beer recipes.

Our results show that deep learning provides a promising approach to the highly non-linear mapping from the domain of recipes to the domain of chemical properties, but there are many future directions. First, it would be worth exploring other encoder architectures that more tightly match the partially ordered nature of brewing recipes. Second, there are many other mappings between domains that would be useful: from chemical attributes to recipe, or between chemical attributes and written beer reviews. Decoder RNNs could employed to produce the recipes and reviews. These extensions would allow brewers to discover novel, optimized recipes, and to more precisely and easily refine existing recipes. Finally, there are richer chemical representations than the ones we had access to in this study; in particular, leveraging gas and liquid chromatography data would be ideal.

ACKNOWLEDGEMENTS

The authors would like to thank the Nvidia corporation for their donations of Titan X and Titan Xp GPUs used in this work.

REFERENCES

[1] Z. C. Lipton, S. Vikram, and J. McAuley, "Capturing meaning in product reviews with character-level generative text models," *CoRR*, vol. abs/1511.03683, 2015. [Online]. Available: http://arxiv.org/abs/1511.03683

[2] C. Carroll, *The Bottlefly IOS Application for Wine Recommendations*. California Polytechnic State University, 2016.

[3] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 897–908. [Online]. Available: https://cs.stanford.edu/people/jure/pubs/beerrec-www13.pdf

[4] C. Kiddon, "Learning to interpret and generate instructional recipes," Ph.D. dissertation, University of Washington, 2016.

[5] C. Kiddon, G. T. Ponnuraj, L. Zettlemoyer, and Y. Choi, "Mise en place: Unsupervised interpretation of instructional recipes," in *Proc. EMNLP*, 2015, pp. 982–992.

[6] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[9] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *EMNLP*, 2013.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[11] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. NIPS*. Curran Associates, Inc., 2017, pp. 3391–3401.

[12] 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/

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

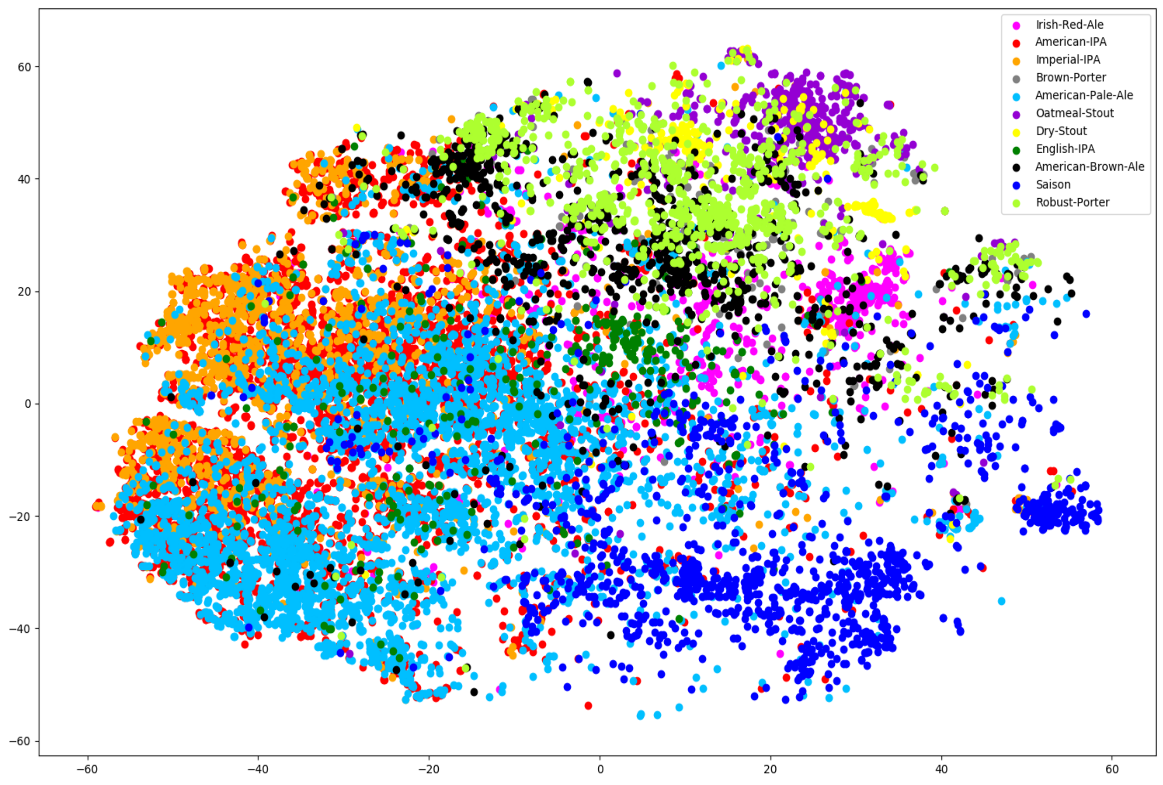

Fig. 3. Plot of LSTM output representation of recipes for the fine-grained type classification task

[16] P. McCullagh, "Generalized linear models," *European Journal of Operational Research*, vol. 16, no. 3, pp. 285–292, 1984.

[17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[18] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. NIPS*, 1996, pp. 155–161.

[19] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Appendix

Below we list the 81 self-reported fine-grained beer types used in our experiments, including one catch-all "Other" category for all types outside of the 80 most common types.

Other, American IPA, Imperial IPA, Sweet Stout, Belgian Pale Ale, Kolsch, Specialty Beer, Weizen/Weissbier, Oktoberfest/Marzen, Dry Stout, English IPA, Special/Best/Premium Bitter, California Common Beer, American Barleywine, Belgian Dubbel, Robust Porter, American Stout, Extra Special/Strong Bitter (English Pale Ale), Saison, Munich Helles, Maibock/Helles Bock, American Pale Ale, Munich Dunkel, Belgian Specialty Ale, Witbier, Oatmeal Stout, American Wheat or Rye Beer, Southern English Brown, Belgian Tripel, Weizenbock, Flanders Brown Ale/Oud Bruin, Russian Imperial Stout, Classic American Pilsner, American Amber Ale, Spice, Herb, or Vegetable Beer, Belgian Golden Strong Ale, Fruit Lambic, Dunkelweizen, Scottish Export 80/-, Biere de Garde, Fruit Beer, Strong Scotch Ale, Cream Ale, Premium American Lager, Blonde Ale, Belgian Blond Ale, Christmas/Winter Specialty Spiced Beer, English Barleywine, Irish Red Ale, Doppelbock, German Pilsner (Pils), Brown Porter, Mild, Roggenbier (German Rye Beer), Bohemian Pilsener, Gueuze, Foreign Extra Stout, Vienna Lager, Traditional Bock, Wood-Aged Beer, Old Ale, American Brown Ale, Dusseldorf Altbier, Standard American Lager, Dark American Lager, Baltic Porter, Other Smoked Beer, Berliner Weisse, Northern English Brown Ale, Standard/Ordinary Bitter, Belgian Dark Strong Ale, Flanders Red Ale, Schwarzbier (Black Beer), Lite American Lager, Scottish Light 60/-, Scottish Heavy 70/-, Straight (Unblended) Lambic, Classic Rauchbier, Northern German Altbier, Dortmunder Export, Eisbock