Unsupervised Learning of Hierarchical Representations for Recipe Text with Deep Belief Networks

Gregory Izatt*

***Abstract*— We explore the application of deep belief networks (DBNs) as language models for the instruction text of cooking recipes. We provide quantitative evidence that DBNs can outperform simpler, non-hierarchical language models, as well as qualitative evidence that they can succeed at learning reasonable hierarchical representations. We also evaluate a method for extracting and improving the hierarchical representations learned by our model, which relies on achieving sparsity in the hidden layers.**[1]

I. Introduction

Cooking is a classic AI task with broad relevance across the fields of natural language processing, knowledge representation, and robotics. While the vocabulary of cooking is more restricted that the broader vocabulary of speech, the extraction of structure from the raw text of a recipe remains a difficult challenge. The internet provides an incredibly large number of recipes, but their natural language representation can be difficult to unravel: coreferences to ingredients and the products of actions abound, and critical steps are often left implicit.

A fair amount of work has been done on the extraction of consistently-formatted, unambiguated recipes from almost-raw text, and almost all of them involve translating existing recipes into graphical representations (see Figure 1). [4] demonstrated that it is possible to assemble these graphs given hand-crafted dictionaries of domain-specific nouns and verbs and appropriate rules to connect them. (These dictionaries can be constructed automatically, however: see, e.g., [14].) This graphical representation translates ingredients and actions into nodes, with directed connections from ingredients or actions into other actions. This representation naturally enforces the implicit structure in a recipe that each ingredient is used exactly once (and then is exhausted), and that the product of each action is used exactly once as either an input to another action, or the final product. It also allows the true process of executing the recipe to be read off relatively easily for execution on a robot (as in, e.g., [3]), and allows comparison to other recipes (by using, e.g., subgraph comparisons [16]). These graphical representations have stuck around, seeing implementation in the International Computer Cooking Contest [15], and even reformulation as a POMDP [8].

However, these graph-construction techniques are constrained by design to represent recipes in a shallow way.

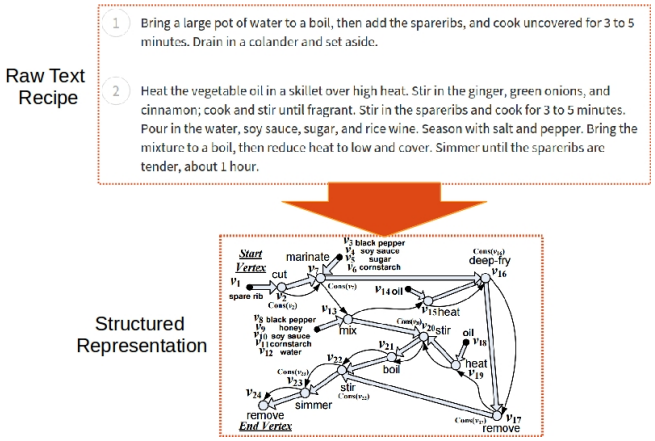

Fig. 1. *Cooking graph* from [16], with example corresponding raw recipe text from *allrecipes.com*.

This has significant benefits, including simplicity of formulation and clarity of evaluation and application of the resulting graph. However, constraining the graphical models to be shallow removes the possiblity of exploiting deeper structure inherent in recipes. Recipes tend to be constructed out of sub-recipes, which constitute common cooking tasks: for example, many baking recipes share the preparation of a ⟨flour⟩+⟨water⟩+⟨leavening agent⟩+⟨activator⟩ dough. Indeed, that these kinds of subrecipes appear as subgraphs was critical to the success of [16]. Such subrecipe information may prove invaluable in parsing more difficult recipes which skip over vital details in the name of ”common sense.” It would then be of significant benefit to explore more hierarchical recipe representations.

Where many of these previous methods are supervised, we focus instead on unsupervised methods for extracting this structure. The state of the art of hierarchical language and topic modeling provides a multitude of options for such unsupervised representation learning, including hierarchical LDA [2], deep belief networks (DBNs) [5], deep Boltzman machines (DBMs) [11], and some mixtures of those models (e.g. hLDA and DBM [12], [1]). DBNs have been applied with moderate success to lists of ingredients standing alone [10], and with great success to many other topics. We explore the application of DBNs to uncover hierarchical topic models for recipes.

*Work performed under the structure of MIT Course 6-864: Advanced Topics in Natural Language Processing, with direction and mentorship from the course staff.

[1]Our code is available for those with MIT certificates at `https://github.mit.edu/gizatt/hierarchical-recipe`.

II. DATASET

We crawled $49,893$ recipes from the internet using the *Scrapy* Python toolbox [13] [2]. For each recipe, the instruction text (without the ingredient list) was parsed into an ordered list of words, which was then prepended with $< START >$ and postpended with $< END >$. **We will refer to each of these parsed recipe instruction vectors as a "sentence," even if they are actually composed of multiple recipe steps or individual grammatical sentences.** From these sentences, containing a total of $5,138,260$ words, we extracted a unique vocabulary of $|V| = 9,400$ words.

To speed development and alleviate technical limitations with our DBN implementation (see Discussion and Future Work), we also created a smaller dataset from the first 2784 recipes in the dataset, with a total of $315,929$ words and a vocabulary size of $|V| = 2784$ words. We refer to the complete dataset as corpus all, and this smaller dataset as corpus $small$.

Each of these corpuses was further split into $train$, $validation$, and $test$ sets, as 93%, 2%, and 5% of their corpus respectively.

III. FORMULATION

Our primary goal is to explore avenues for learning and extracting hierarchical representations for the highly structured text of cooking recipe instructions. We tackle this by applying a DBN as a language model for such a corpus, with the hope that in learning a model for the language, it will arrive on such a representation due to the particular structure of DBNs. This method has shown success in the past: for example, [6] demonstrated that DBNs arrive on hierarchical representations for visual inputs very similar to those identified in the visual area V2, and [7] extended this work with additional convolutional structure to uncover a hierarchy of features for face recognition [7].

DBNs are a subclass of probabilistic graphical models, formed by stacking bipartite Markov random fields known as Restricted Boltzmann Machines (RBMs). [5] demonstrated that they can be trained by greedily training each RBM from the bottom up to behave as an autoencoder using a gradient approximation technique (contrastive divergence, CD). We can use the DBN model to estimate $P(\mathbf{w})$ for an input sentence $\mathbf{w}$. To do this, we consider the input to our DBN model to be a bag-of-words: a binary vector of the size $|V|$, where we set the i^{th} input node to 1 if the i^{th} word in the vocabulary appears in $\mathbf{w}$. We can use Gibbs steps to sample from the DBN the probability of each visible node being active given that initial condition, and then compute the cross-entropy between that multivariate distribution and the original bag-of-words computed from the sentence. During training, we use 5 Gibbs steps for CD calculation. During sentence prediction, we use 1 Gibbs step to calculate bag-of-word probabilities.

[2]Code available at `https://github.com/gizatt/scraping_samples`, though raw data is included in the main project repository.

The work of [6] suggests that the addition of a cost penalty encouraging sparsity among the nodes of the hidden layers of the DBN can encourage the network to arrive at better hierarchical models – or, at the very least, make those models easier to extract. This strategy makes sense, from the observation that without such a penalty, it is likely that even if there is some mapping of hidden node values to learned hierarchical topics, a given topic is likely to be distributed across nodes, and hence difficult to discover. We experiment with the addition of a similar sparsity penalty: in between every CD-gradient step, we used a gradient descent step to descend a cost function that penalized deviation of the $L\frac{1}{2}$-norm across the $|\mathbf{h}|$ hidden units of each RBM from $p = \frac{|\mathbf{h}|}{10}$:

$$\min \left| p - \left(\sum_{h \in \mathbf{h}} |\mathbf{E}[h_l | v_l]|^{\frac{1}{2}} \right)^2 \right|$$

We consider the comparison of a single-layer DBN (i.e. a single RBM) to a two-layer DBN (see Figure 2), each with and without the $L\frac{1}{2}$ penalty. We implement these DBNs in Python, leveraging $numpy$, $scipy$, $scikitlearn$, and $Theano$[3].

IV. EVALUATION

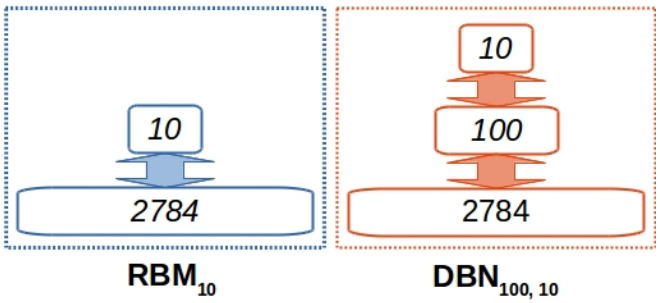

Fig. 2. DBN architectures we compare. *Left:* A single-layer DBN, consisting of a single RBM with $|V| = 2784$ visible nodes and 10 hidden nodes. *Right:* A two-layer DBN, mapping a $|V| = 2784$-node input through hidden layers of 100, and then 10, nodes.

As we are learning a language model, we can adopt *perplexity* as a quantitative measure corresponding roughly to the compressive ability of the model. If our language model predicts the probability of a sentence $\mathbf{w}$ is $P_M(\mathbf{w})$, then we define the per-word perplexity:

$$2^{-\frac{1}{N} \sum_{\mathbf{w} \in Corpus} \log_2 P_M(\mathbf{w})}$$

where N is the total number of words in the corpus.

V. BASELINES

To provide a reference on our quantitative measure, we implement a few baselines to judge performance against:

[3]`http://deeplearning.net/software/theano/`

A. Uniform

If we consider all words in the vocabulary to be equally likely, then we can assign $P_{uniform}(w \in V) = \frac{1}{|V|}$. Then we expect $P_{uniform}(\mathbf{w}) = P_{uniform}(w_1 w_2 ... w_k) = \Pi_{i=0}^{k} P_{uniform}(w_i) = \frac{1}{|V|}^k$, with a per-word perplexity equaling the vocabulary size.

B. Unigram

If we factor all sentence probabilities such that words are considered to occur independently, then we see that $P_{unigram}(\mathbf{w}) = P_{unigram}(w_1 w_2 ... w_k) = \Pi_{i=0}^{k} P_{unigram}(w_i)$ can be maximized over a corpus by learning $P_{unigram}(w)$ for all $w \in V$ as ratio of occurrences of w in the corpus versus the occurrences of all words. This is the classic unigram model. We include add-ϵ smoothing to account for words missing in the training set but appearing in validation or test. We chose $\epsilon = 1$ by empirical evaluation on the validation set.

C. Bigram

Similarly, if we factor sentence probabilities such that words depend only on their predecessors, we see that $P_{bigram}(\mathbf{w}) = P_{bigram}(w_1 w_2 ... w_k) = \Pi_{i=0}^{k} P_{bigram}(w_i | w_{i-1})$ can be maximized over a corpus by learning the bigram occurrences of w_{i-1}, w_i for all word pairs in the corpus. This is the classic bigram model. We include add-ϵ smoothing here as well. We chose $\epsilon = 0.005$ by empirical evaluation on the validation set.

D. Recurrent Neural Network

Out of interest in experimenting with classes of recurrent neural networks (RNNs) that have gained significant popularity in recent years, we include an initial exploration of RNNs applied to this language modeling task, leveraging $theanets$[4] to decrease developmental overhead. The RNN architecture we explore uses a $|V|$-node one-hot input layer, a 100-node tanh encoding layer, a 100-node LSTM layer, and a $|V|$-node softmax output layer. It is trained to predict the probability of the next word in a sentence given the current word, after being exposed to each of the previous words in the sentence in sequence. The RNN is trained by backpropagation of cross-entropy prediction error.

E. Latent Dirichlet Allocation

We also provide a framework for applying Latent Dirichlet Allocation (LDA) to this dataset, by leveraging $scikitlearn$. Given a corpus, LDA determines a set of topics that explain the co-occurrence of sets of words in the sentences of the corpus. This topic modeling is conceptually similar to what is performed by a single RBM, if the nodes of the RBM (or perhaps combinations of them) represent topics; hence we hoped that this baseline would offer a good basis for comparison with our DBN. We use $n = 10$ topics.

[4] https://github.com/lmjohns3/theanets

VI. QUANTITATIVE EVALUATION

Model performances on the the $small$ and all corpuses are reported in Table 3. All results are computed on the held-out test subset of the corresponding subset, which was novel to the model.

Performance on the Uniform, Unigram, and Bigram baselines are unsurprising, and provide a sanity check concerning what perplexity values we should expect from better models.

RNN performance was somewhat disappointing: while the RNN should have been able to outperform the bigram baseline (at the very least), we could not tune it to near that level. While we had time to test a handful of variants on our RNN architecture, that search did not improve the quality of the model. Preliminary results from an unconverged model trained on the all corpus also suggests that simply training on more data will not improve the performance. We suspect a hyperparameter problem, but did not have time to pursue the issue due to focus on analyzing and improving the DBN model.

LDA performance was also disappointing, though for a different reason. LDA dramatically overfit on the smaller corpus, and still fairly strongly overfit on the larger corpus. This overfitting, and generally weak performance (barely better than unigram!) implies the need for better regularization or smoothing; however, those were knobs we did not have time to tune due to focus on analyzing and improving DBN model.

In contrast, DBN performance is exceptionally good – so good that wish to present these quantitative results only cautiously and tentatively, out of fear of mistakes in our implementation. Taking our results at face value, the RBM, with or without sparsity, was the best performing model by a wide margin, with the two-layer DBN following behind. The sparsity did not appear to have a significant impact on the ultimately quantitative performance, though we would need to train many more DBMs to be sure. We hypothesize this may be due to the relatively small architectures, and hence relatively simple search spaces. That the RBM outperforms the deeper model is counter to our hypothesis: we expected extra depth to prove useful to the model. We think, in this case, the opposite occurred: the extra layer added a significant number of parameters to the model beyond what our small dataset could support, resulting in overfitting and a worse overall model.

There are several explanations for the performance disparity between the baselines and the DBNs. The first is that we did not have adequate time to tune the nontrivial baselines (LDA and RNN) to a sufficient degree to match the performance of the DBN. The second is that we are, to a degree, comparing apples to oranges: the DBN is calculating sentence probabilities based on a bag-of-words model, while other baselines are relying on the temporal ordering of words to calculate probability. We contend that the same probability – the probability of a sentence being a member of the language of recipes – is being calculated in both cases, but the means that the model uses to calculate the probability

(and hence the complexity of the model being learned) is more complicated in the case of the worse-performing baselines.

These low perplexities certainly indicate the remarkable structure inherent in recipe texts. That recipe texts are so highly structured is not entirely surprising; but that such a small RBM is able to capture that structure is certainly impressive.

Model	Perplexity	
	small	all
$Uniform$	3722	9400
$Unigram$	334.96	346.93
$Bigram$	54.12	36.02
RNN	161.04	—-
LDA	1799.33	387.06
$LDA,\ on\ train$	293.29	275.69
RBM_{10}	**4.79**	—-
$RBM_{10}, Sparse$	**4.75**	—-
$DBN_{100,10}$	**7.40**	—-
$DBN_{100,10}, Sparse$	**7.37**	—-

Fig. 3. Perplexities across both corpuses, and all language models, where available. Perplexity performance is not available on the *all* corpus due to time and memory constraints on the RNN, RBM, and DBN models. Save on the second LDA statistic, all performances are estimated over the withheld test set; the second LDA statistic reports LDA perplexity estimated over the training set. DBN performances are the average of three separately trained models, initialized with different random seeds; the performances of the individual models are listed in the README of the code respository.

VII. QUALITATIVE INVESTIGATION OF THE MODEL

If we hypothesize that the way the DBN is storing the input bag-of-words among its hidden nodes is by discovering a number of topics, which are present when a given hidden node (or combination thereof) is active, then we ought to be able to unveil those topics by performing the reverse operation. That is, we can turn on a set of hidden nodes and propagate activation downward to see which words most strongly correlate with those nodes. (It makes more sense that this technique should work when the sparsity penalty is applied; however, we'll see that it seems to work even without enforcing such additional structure.) To generate each bag-of-words shown here, we set a single hidden node in the target layer to 1; propagate activations downwards according to the learned weights of the model; and select the top words indicated by the mean values of the nodes of the visible layer. Words are displayed in decreasing order of likelihood.

A. RBN_{10}, Top Layer

Activating single hidden nodes of RBM_{10} reveals that individual nodes correspond strongly to distinct bags-of-words with thematic similarity. In a representative run of both sparse and nonsparse RBM_{10}, there were no common words other than the $< START >$ and $< STOP >$ symbols shared among any of the discovered bags-of-words. Further, all of the bags-of-words had at least one common theme (though it appeared that they may have had multiple common themes each).[5] One such bag-of-words from the best performing sparse DBN_{10} included a distinct dessert flair:

> NODE 0: and bake sugar beat confectioners vanilla mixer cake electric f cookies degrees wire speed extract fluffy stiff dough rack beating pie crust after oven peaks or cookie whites chocolate eggs soda cool batter each whipped filling cream condensed top press mixing sift ingredients sweetened cup inch pans 325 racks c addition ice bottom light flour sheets onto 9 ungreased cocoa greased frosting shortening creamed egg spread 1 cinnamon gradually glass manufacturer of at frost almond parchment freeze graham chips will white form round icing floured butter cakes topping shell chill milk make time cooling edges pecans metal combine ...

B. $DBN_{100,10}$, Top Layer

Activating single hidden nodes of the top layer of $DBN_{100,10}$ revealed qualitatively similar behavior, though the topics tended to be less well-defined. [6] In both sparse and nonsparse DBN, many more words were shared between the each of the high-level nodes. After removing those shared words, the remaining nodes had, perhaps half of the time, discernable theme. For example, the best-performing non-sparse $DBN_{100,10}$ had a distinct "cake" high-level neuron:

> NODE 5: and until in beat smooth medium vanilla flour sugar milk frosting about cake electric mixture all egg bowl add cream out cocoa into over confectioners vegetable before butter water creamy icing combined food 5 well grease powder is yogurt extract mixer ingredients softened soda cook pans skillet each pan whites fill bananas eggs beating with between stirring speed high margarine place clean batter chilled after top use or at gradually separate reached then of blend banana round coconut frost 3 light constantly sift ...

However, many other nodes corresponded to bags of generic common recipe words. For example:

> NODE 7: and in degrees medium into over until skillet of c 5 top about f place with bowl serve mixture cream egg baking heat each cook at an on side milk smooth ground ice garlic 350 sauce per sheet wrap preheat 3 preheated form or hot pour high bake pepper 30 eggs out slice should plastic dip before once sides center spoon water down pan all is half 4 powder your food more low onion constantly oven dish then black well toasted set oil ...

C. $DBN_{100,10}$, Middle Layer

Of great interest to us in our pursuit for *hierarchical* representation is whether this method, applied to the *middle* layer of hidden nodes, reveals that the network has discovered substructure of recipe topics.[7] We find that, in both the sparse and nonsparse $DBN_{100,10}$, the intermediate nodes correspond to a mixture of mostly similar bags of vaguely related recipe words:

> NODE 68: all 2 of food using inch after 1 on not paper onto into is at knife up from or bottom add seal if inches press that sugar each they air around necessary it surface process mixture filling water large lid for spatula least rack make the an you moist with top tops store cool out together bubbles 4 move about filled tight lower have are within towel jars by pour lids will so apart may screw place down rolling until fill thin

[5] All 10 bags-of-words with minor commentary available in the project repo at `dbn/small_10_l12_sparse_0001.dbn.samples` for the sparse RBM, and `dbn/small_10_nonsparse_0002.dbn.samples` for the nonsparse RBM.

[6] All 10 bags-of-words with minor commentary available in the project repo at `dbn/small_100_10_l12_sparse_0002.dbn.samples` for the sparse DBN, and `dbn/small_100_10_nonsparse_0002.dbn.samples` for the nonsparse DBN.

[7] Intermediate node bags-of-words available in the same files as listed for the top-level bags of words for each DBN.

carefully leave rims use halfway salt remove as enough flour off several area mix sterilize bowl ...

However, we found a handful of much more specific nodes corresponding to sensible cooking substructures. For example, the best-performing sparse $DBN_{100,10}$ produces a distinctive node corresponding to the act of testing whether a baked good is done with a toothpick, with the top 6 most likely words corresponding to that action:

NODE 40: **out inserted clean comes toothpick center** bowl using knife of cake large time into fluted banana confectioners r until alternating or preheated all cupcakes creamy cupcake ...

A very similar node is present in the nonsparse DBN, perhaps indicating the stability of this behavior:

NODE 35: **inserted toothpick comes clean** frosting cake **out** boil s saucepan two **center** simmer pans batter bring after icing manufacturer tube beating addition an cocoa ...

Another intriguing substructure relates to the presence of rice krispies in a handful of recipes in the corpus:

NODE 15: and firm crust manufacturer maker s boil cracker krispies graham c buttered kellogg salted using freeze condensed ice decorate cream marshmallows best cereal instructions shortening sweetened a ...

VIII. DISCUSSION AND FUTURE WORK

That the RBM and DBN beat the baselines confirm that recipe instructions have extremely significant inherent structure on a scale beyond what our relatively simple and local baselines could take advantage of. Further, our qualitative investigation of the RBM and DBN suggests the models are truly relying on sensible and significant topic-based structure within the corpus. The RBM appears to perform similarity to a good topic model: it forms a basis of topics to span the space of recipes, with each topic corresponding some number of words that tend to co-occur. The DBN – the deeper model – shows hints of uncovering the kind of *hierarchical* topical structure we hoped for, in representing more specific recipe subparts and substructures with its intermediate nodes.

However, significant work remains to be done concerning many aspects of this work. Perhaps the clearest future work entail investigating the lackluster performance of our baselines. While the failure of the RNN can perhaps be chalked up to an improper architecture, implementation mistake, or hyperparameter choice, the extremely poor performance of the LDA is very surprising, considering the simplicity of the corresponding code, and merits investigation.

Additionally, it contradicts our hypothesis that the DBN is outperformed by the shallower RBM, particularly considering the very small number of hidden nodes available to the RBM. We suspect this is due to the additional difficulty in training the larger DBN, along with a mismatch between the size of our corpus, and the complexity of our models. The DBN model is sufficiently high-dimensional that using a dataset as small as we have is risky, and is likely to have caused significant overfitting.[8]

[8]The reason we were restricted to using less than 10% of our available corpus for training is simple but resolvable: our original Theano implementation of DBN was not created with loading in different sections of a massive corpus during training, instead opting to load the entire corpus simultaneously to GPU memory. By the time it was clear that corpus size was limiting our performance, it was too late to make significant architectural modifications to the codebase.

In formulating DBNs as considering the likelihood of bags-of-words as opposed to ordered lists of words, we discarded temporal information that is critical in the formulation of implementable cooking graphs. While this significant compromise was necessary to keep our work within a reasonable scope, there is a way forward: application of the convolutional DBN structure advocated in [7] could allow us to recover temporal structure without blowing up the input dimension size of the network.

Finally, on a more fundamental level, our method of extracting the learned structure from DBNs is not general or automatic enough to be extensible to larger DBNs, or to allow extraction of directly practical recipe structures like the cooking graphs of [4]. While it was our hope that adding a sparsity penalty might lead to a significant additional structuring to the representation used by the DBN (in terms of forcing it to use only a small number of nodes to encode a given recipe), the penalty we applied did not have measureable impact on either quantiative or qualitative results.

In conclusion, we have demonstrated hints that DBNs can recover hierarchical structure from recipe instructions in an unsupervised manner. However, the development of means for improving, extracting, and applying that structure remains an open problem.

ACKNOWLEDGMENTS

My thanks to Regina Barzilay and Tommi Jaakkola for their direction early in the project. Particular thanks to Karthik for his specific advice on improving both my RNN and DBN implementations. My thanks to you three, as well as Franck and Tianheng, for a great term!

REFERENCES

[1] Yoshua Bengio, Aaron Courville, Pascal Vincent, Representation Learning: A Review and New Perspectives (2013), *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), pp. 1798-1828.

[2] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, Joshua B. Tenenbaum Hierarchical Topic Models and the Nested Chinese Restaurant Process (2012), *Neural Information Processing Systems 16*.

[3] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, Daniela Rus, Interpreting and Executing Recipes with a Cooking Robot (2013), *Proceedings of ISER*.

[4] Reiko Hamada, Ichiro Ide, Shuichi Sakai, Hidehiko Tanaka, Structural Analysis of Cooking Preparation Steps in Japanese (2000), *Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages*, pp. 156-164.

[5] Geoffrey Hinton, Simon Osindero, Yee-Whye Teh, A fast learning algorithm for deep belief nets (2006), *Neural Computation*, 18, pp. 1527-1554.

[6] Honglak Lee, Chaitanya Ekanadham, Andrew Y. Ng, Sparse deep belief net model for visual area V2 (2007), *Proceedings of the 29th Conference on Neural Information Processing Systems*.

[7] Honglak Lee, Roger Grosse, Rajesh Ranganath, Andrew Y. Ng, Convolutional Deep Belief Networks for Scalabe Unsupervised Learning of Hierarchical Representations (2009), *Proceedings of the 26th International Conference on Machine Learning*.

[8] Jon Malmaud, Earl J. Wagner, Nancy Chang, Kevin Murphy, Cooking with Semantics (2014), *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pp. 33-38.

[9] Tomas Mikolov, Kai Chen, Greg S. Corrado, Jeffrey Dean, Efficient estimation of word representations in vector space (2013), *International Conference on Learning Representations*.

[10] Vladimir Nedovic, Learning recipe ingredient space using generative probabilistic models (2013), *Proceedings of the Cooking with Computers Workshop*, 1, pp 13-18.
[11] Ruslan Salakhutdinov, Geoffrey Hinton, Deep Boltzman Machines (2009), *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, 12.
[12] Ruslan Salakhutdinov, Joshua B. Tenenbaum, Antonio Torralba, Learning with Hierarchical-Deep Models (2013), *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35.
[13] http://scrapy.org/
[14] Shunsuke Mori, Tetsuro Sasada, Yoko Yamakata, Koichiro Yoshino, A Machine Learning Approach to Recipe text Processing (2012), *Proceedsing of Cooking with Computers Workshop*.
[15] Kristin Walter, Mirjam Minor, Ralph Bergmann, Workflow Extraction from Cooking Recipes (2011), *Proceedings of the ICCBR 2011 Workshops*, pp. 207-216.
[16] Liping Wang, Qing Li, Na Li, Guozhu Dong, Yu Yang, Substructure Similarity Measurement in Chinese Recipes (2008), *WWW 2008, ACM*, pp. 979-988.