Learning to refine text based recommendations

Youyang Gu and **Tao Lei** and **Regina Barzilay** and **Tommi Jaakkola**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{yygu,taolei,regina,tommi}@csail.mit.edu

Abstract

We propose a text-based recommendation engine that utilizes recurrent neural networks to flexibly map textual input into continuous vector representations tailored to the recommendation task. Here, the text objects are documents such as Wikipedia articles or question and answer pairs. As neural models require substantial training time, we introduce a sequential component so as to quickly adjust the learned metric over objects as additional evidence accrues. We evaluate the approach on recommending Wikipedia descriptions of ingredients to their associated product categories. We also exemplify the sequential metric adjustment on retrieving similar Stack Exchange AskUbuntu questions. [1]

1 Introduction

Modern recommender problems involve complex objects, often described in textual form. In order to learn to predict how disparate objects may go together, it is helpful to first map them into a common representation where they are easily compared, regardless of their origin. Neural models are particularly well-suited for this task as continuous vector representations of objects can be tailored in a flexible way to the desired task. While these models have been shown to be effective across NLP tasks (Sutskever et al., 2014; Andreas et al., 2016; Hermann et al., 2015), they take considerable time to learn and are therefore ill-suited to be adjusted rapidly as additional evidence accumulates.

We cast our text-to-text recommendation problem in two phases. In the first phase, flexible neural text-to-vector mappings are learned from currently available data. Such mappings are optimized to function well in a collaborative filtering setting. For example, in the context of recommending food product categories for ingredients based on their Wikipedia pages, the continuous vectors are adjusted so that their inner product directly reflects the degree of association between the objects. Once learned, the mapping can be applied to any previously unseen text to yield the corresponding vector representation, and therefore also used for predicting associations. In the second phase, we no longer adjust text-to-vector mappings but rather parameterize and learn how the vectors are compared. For example, we can optimize the metric separately for each new ingredient based on a few category observations for that ingredient. The goal of this second phase is to specifically boost the accuracy when the neural baseline (unaware of the new evidence) would otherwise not perform well.

Our approach builds on the recent work on recurrent convolutional models to obtain text-to-vector mappings (Lei et al., 2015; Lei et al., 2016). This architecture is particularly well suited for noisy Wikipedia pages as it can learn to omit and highlight different parts of the text, as needed. The additional sequential component is a regularized logistic regression model (for ingredient-product prediction) or a ranking model (for question retrieval). We demonstrate the accuracy of the baseline neural recommender and the gains from the second sequential phase in both of these tasks.

[1] The code/data is available at https://github.com/youyanggu/rcnn.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2103–2108,
Austin, Texas, November 1-5, 2016. ©2016 Association for Computational Linguistics

2 Related Work

A great deal of recent effort has gone into developing flexible neural models for text and their use across variety of NLP tasks. This includes building vector representations for sentences and documents (Le and Mikolov, 2014), convolutional neural network models of text (Collobert and Weston, 2008; Zhang and LeCun, 2015), non-consecutive variants of CNNs (Lei et al., 2015), and compositional architectures (Socher et al., 2013), among many others. Our work is most closely related to the use of such models for question retrieval (Lei et al., 2016) but differs, in particular, in terms of our two-phase collaborative filtering formulation and the ingredient mapping task from Wikipedia pages (cf.(Sutskever et al., 2011; Song and Roth, 2015)).

3 Recommender Problems

We explore two recommender problems in this work. In the first problem, we are given a food ingredient, and our goal is to predict which product categories it could appear in. Both ingredients and product categories are provided in terms of natural language descriptions via their associated Wikipedia pages. For example, if given "tomato", we would predict "canned foods" as one likely category for the ingredient. A small number of categories appear as targets for each ingredient.

We also consider the task of predicting questions that are similar to the one provided as a query. The purpose is to facilitate effective question answering by retrieving related past questions (and the associated answers that are available). For this we use Stack Exchange's AskUbuntu question retrieval dataset used in recent work (dos Santos et al., 2015; Lei et al., 2016)

4 Approach

We explain our approach in terms of the first task: predicting product categories from ingredients. Collaborative predictions are made by mapping each ingredient into a vector representation and comparing that representation with an analogous one for product categories. We train these vectors in an end-to-end manner to function well as part of the collaborative task. The vector representations are based on Wikipedia pages that are available for most ingredients and categories in our problem. Rather than derive the vector from the entire article (which can be long), we only use the top summary section. For the AskUbuntu question-answering dataset, we make use of both the title and the question body.

We use a recurrent neural network (RNN) model to map each text description into a vector representation. Our model builds on the recurrent convolutional neural network model of (Lei et al., 2016) used to train the AskUbuntu question representations. We describe below a modified version used for ingredient-product category prediction.

Let $v_\theta(x) \in \mathbb{R}^d$ be the parameterized RNN mapping of text x into a vector representation, where d is the dimension of the hidden representation. Let x_i and z_p be the Wikipedia pages for ingredient $i \in I$ and product category $p \in P$, respectively. We use the same parameters θ to generate the representations for both ingredients and product categories due to their overall similarity. Thus $v_\theta(x_i)$ is the vector representation for ingredient i and $v_\theta(z_p)$ is the vector representation for product category p for an RNN model with parameters θ. We train the RNN model to predict each association $Y_{ip} = 1$ as a binary prediction task, i.e.,

$$P(Y_{ip} = 1|\theta) = \sigma(v_\theta(z_p) \cdot v_\theta(x_i)), \qquad (1)$$

where σ is the sigmoid function $\sigma(t) = (1 + \exp(-t))^{-1}$. The formulation is akin to a binary collaborative filtering task where user/item feature vectors are produced by the RNN. The parameters θ can be learned by back-propagating log-likelihood of the binary 0/1 predictions back to θ.

4.1 Sequential learning

Our RNN model, once trained, will be able to map any new ingredient and product category (their text descriptions) into vectors, and make a binary prediction of whether the two go together. However, training the model takes considerable time and cannot be easily adjusted in the face of new evidence, e.g., a few positive and negative categories for a previously unseen ingredient. Since RNN features are global (affecting the mapping from text to features for all ingredients/products), it is not clear how the adjustments made in light of additional information about a specific new ingredient will impact predictions for other ingredients. We propose a sequential approach that is instead local, tailored to the new ingredient.

In order to sequentially adjust the model predictions with new evidence, we introduce parameters $w = [w_1, \ldots, w_d], w_j \in \mathbb{R}^+$ that modify the comparison of ingredient and category vectors. Specifically, the association is predicted by

$$P(Y_{ip} = 1|\theta, w) = \sigma\{v_\theta(z_p)^T \text{diag}(w) v_\theta(x_i)\}, \tag{2}$$

where diag(w) is a diagonal matrix with the entries specified by w. We assume that, at this stage, the RNN parameters θ and therefore the vector representations $v_\theta(z_p)$ and $v_\theta(x_i)$ are nonadjustable. We will only update weights w in response to each new observation, separately for each ingredient. The observations can both be positive ($Y = 1$) and negative ($Y = 0$).

Because we expect a new input may only have a small number of observations, it is important to properly regularize the weights as to avoid overfitting. We append the log-likelihood objective with a regularizer

$$reg(w) = \frac{\lambda}{2} \sum_{j=1}^{d} (w_j - 1)^2 \tag{3}$$

where λ is the overall regularization parameter. Note that for large values of λ, the regularizer keeps the parameters at the default values $w_j = 1$ corresponding to the baseline RNN collaborative predictions, unmodified by the new evidence.

In the context of predicting similar questions, we use a modified binary formulation where the goal is to classify each triplet of questions (x, z_1, z_2) in terms of whether z_1 is closer to the query than z_2. In this ranking model, the probability that z_1 is closer is given by

$$\sigma\Big((v_\theta(z_1) - v_\theta(z_2))^T \text{diag}(w) v_\theta(x)\Big), \tag{4}$$

The parameters w are again trained from observed additional triplet relations in the AskUbuntu dataset. The parameters w are regularized as in the ingredient-product category setup.

The sequential part can therefore be viewed as a content recommendation task which is tailored to the specific query (e.g., ingredient) using features from previously trained RNNs. It assumes additional feedback in order to adjust the feature comparison using the introduced weights w.

5 Experimental Setup and Results

Ingredients: We use the *FoodEssentials LabelAPI*[2] and *Rapid Alert System for Food and Feed (RASFF)*[3] databases to extract 5439 ingredients and the product categories they appear in. On average, each ingredient appears in 16.3 product categories (out of 131 categories). We leverage Mechanical Turk to link each ingredient to the appropriate Wikipedia article. From the 5439 ingredients, there are 1680 unique Wikipedia articles. Each ingredient summary description has a median of 169 tokens.
AskUbuntu: The dataset consists of 167k questions and 16k user-marked similar question pairs taking from a 2014 dump of AskUbuntu website.

5.1 Training, development, and test sets

Ingredients: We take the set of unique Wikipedia articles and randomly split them into training, development, and test sets (60/20/20). We then assign the ingredients to the appropriate data set based on their Wikipedia articles. This is to ensure that the articles of the ingredients used in the development and test sets are not seen in training.
AskUbuntu: We take 8000 human annotated question pairs as our development and test sets. There are 200 query questions in each set. Each query question is paired with 20 candidate questions which are annotated as similar or non-similar. We evaluate by ranking these candidate questions.

5.2 Sequential scenario

Ingredients: Let n be the total number of labeled positive categories for the ingredient. We provide $min(20, n/2)$ positive categories for the sequential model to train. We also include k negative categories, where k is selected using the validation set. We evaluate the performance on the remaining $n - min(20, n/2)$ positive categories as well as on the negative categories not included in training.

[2] http://developer.foodessentials.com/

[3] http://ec.europa.eu/food/safety/rasff/index_en.htm

Ingredient	*Wikipedia article*	*Prediction 1*	*Prediction 2*	*Prediction 3*
oatmeal	Oatmeal	cereal (0.564)	snack, energy & granola bars (0.196)	breads & buns (0.039)
watermelon juice	Watermelon	fruit & vegetable juice (0.352)	ice cream & frozen yogurt (0.205)	yogurt (0.064)
jasmine rice	Jasmine rice	flavored rice dishes (0.294)	rice (0.237)	herbs & spices (0.062)
shrimp extract	Shrimp (food)	fish & seafood (0.491)	frozen dinners (0.128)	frozen appetizers (0.113)
meatball	Meatball	pizza (0.180)	breakfast sandwiches (0.128)	frozen dinners (0.120)
polysorbate 80	Polysorbate 80	chewing gum & mints (0.531)	candy (0.092)	baking decorations (0.049)
ketchup	Ketchup	ketchup (0.461)	salad dressing & mayonnaise (0.049)	other cooking sauces (0.044)
benzoic acid	Benzoic acid	powdered drinks (0.062)	fruit & vegetable juice (0.051)	candy (0.045)

Table 1: The three most likely food product category predictions generated by the baseline RNN model on eight unseen ingredients. The number in parenthesis represents the probability provided by the model.

AskUbuntu: We use the difference vectors in Equation 4 to compute the loss and sequentially update the feature weights w. Let n be the total number of labeled positive examples (similar questions). We select up to $n/2$ positive and negative examples. From the $n^2/4$ possible pairs, we select the 20 most informative pairs for training.

While we use the loss function commonly used for binary classification during training, we ultimately want to frame our question as a ranking problem. Therefore, after iterating through the initial observations, we compute the mean average precision (MAP) over the remaining (unseen) ingredients/questions and compare it to the MAP of the baseline RNN model on the same unseen examples.

5.3 Hyperparameters

RNN: We use Adam (Kingma and Ba, 2015) as the optimization method with the default setting suggested by the authors. We use a hidden dimension of $d = 50$ for the ingredients and $d = 400$ for the AskUbuntu questions. Additional parameters such as dropout (Hinton et al., 2012), hidden layers, regularization, stopping criteria, batch size, and learning rate is tuned on the development set.

Word Vectors: For the ingredient/product prediction task, we used the GloVe pre-trained vectors (Common Crawl, 42 billion tokens, 300-dimensional) (Pennington et al., 2014). The word vectors for the AskUbuntu vectors are pre-trained using the AskUbuntu and Wikipedia corpora.

Sequential: We utilize the bounded limited-memory BFGS algorithm (L-BFGS-B) (Byrd et al., 1995) to solve for the optimal feature weights with bounds $w_j \in [0.01, 2]$. We tuned the the constraint bounds and the regularization parameter λ on the development set.

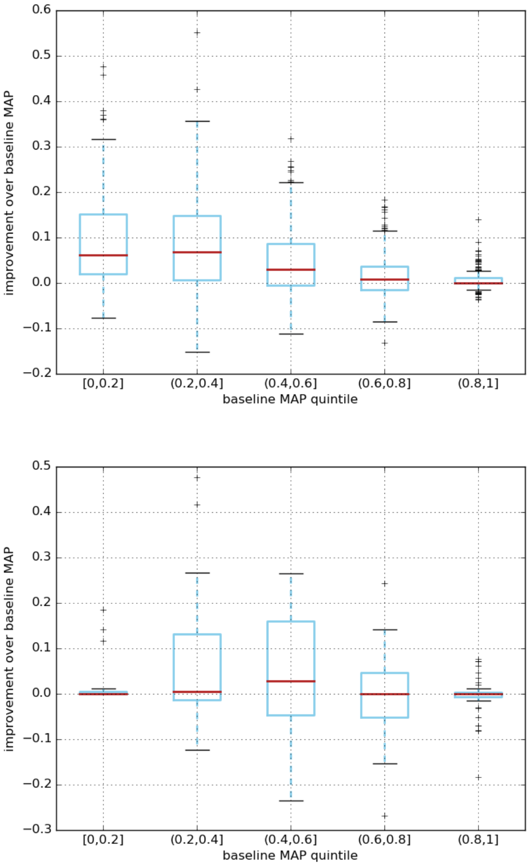

Figure 1: Box plot of the mean absolute mean average precision (MAP) improvement of the sequential model on the ingredients dataset (top) and AskUbuntu questions (bottom). They are divided into five quintiles based on the baseline RNN MAP score. The model shows gains in cases where the baseline RNN model's performance is poor or mediocre. The number of data points in each of the five quintiles of the ingredients dataset are, respectively: 131, 210, 240, 135, 191. For the AskUbuntu dataset, they are: 15, 26, 32, 40, 41.

	Ing / Dev	Ing / Test	AskUbuntu / Dev	AskUbuntu / Test
Mean MAP gain (percent)	0.0525 (30.9%)	0.0492 (26.5%)	0.0246 (8.2%)	0.0224 (7.5%)
Mean # positive observations	8.6	9.1	3.2	2.9

Table 2: We show the mean absolute improvement in the mean average precision (MAP) over the unobserved data points for each ingredient/question. The percent improvement shown is an average percent improvement across the ingredients/questions. They are the average of 100 runs per ingredient and 20 runs per AskUbuntu question.

Model	Validation set	Test set
Random	0.150 / 0.120	0.158 / 0.129
Baseline	0.320 / 0.291	0.331 / 0.300
MLP	0.432 / 0.390	0.459 / 0.416
RNN	**0.476 / 0.422**	**0.478 / 0.426**

Table 3: Results of the RNN model on the ingredient dataset, averaged across 5 runs. The two metrics shown are the mean average precision (MAP) and precision at N (P@N), where N is the total number of positive examples. The random model generates a random ranking of food categories for each ingredient. The baseline model uses the mean occurrence distribution of the food categories for all ingredients to rank the predictions. The multilayer perceptron model (MLP) is a three-layer neural network trained on the hierarchical properties of the input ingredients (extracted from the UMLS Metathesaurus). The RNN model outperforms all other baselines.

5.4 Results

Table 1 and 3 shows our results from using RNN to predict likely food product categories from Wikipedia text descriptions of ingredients.

We show the gains of the sequential update model in Table 2. We are able to generate consistent improvements in the MAP after seeing half of the observations. Box plots of the test set MAP improvements can be seen in Figure 1. For the ingredients prediction task, the sequential model offers the greatest improvements when the baseline RNN has low MAP. In the AskUbuntu questions, on the other hand, the positive effect is greatest when the baseline MAP is around 0.5.

There are three possible reasons for the difference in performance between the two tasks:

- The mean number of positive observations in the AskUbuntu task is 2.9, compared to 9.1 observations in the ingredients task (Table 2). This is a key factor in determining the sequential model's ability to tune for the optimal parameters. Having access to more annotated data would likely result in an increase in performance.
- Owing to the complexity of information encoded, the vectors for the AskUbuntu task are of dimension of 400 as opposed to 50 in the ingredients task. As a result, the sequential model would require more feedback to find near optimal weights w.
- We hypothesize that the sequential model leads to the most increased performance when the baseline model is mediocre. This is especially highlighted in the AskUbuntu task, as extremely poor performance indicate a complete mismatch of questions, while an exceptional performance leaves little room for additional improvement.

6 Conclusion

We demonstrated a text-based neural recommender approach to predict likely food products from a given ingredient as well as other similar questions from a given AskUbuntu question. We then extended this model to an online stream of new data, which improves over the off-line trained version for both of the two tasks tested. This sequential process improves model performance while requiring minimal additional training time and resources.

7 Acknowledgments

We thank the MIT NLP group and the reviewers for their helpful comments. The work was partially supported by the U.S. Food & Drug Administration, and by Google Faculty Award (Barzilay and Jaakkola). Any opinions, findings, conclusions, or recommendations expressed in the paper are those of the authors alone, and do not necessarily reflect the views of the funding organizations.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *International Conference on Machine Learning (ICML 2008)*.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grenfenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NIPS 2015)*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representation (ICLR 2015)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *International Conference on Machine Learning (ICML 2014)*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2015)*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. 2016. Semi-supervised question retrieval with recurrent convolutions. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Yangqiu Song and Dan Roth. 2015. Unsupervised sparse vector densification for short text similarity. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2015)*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural network. *Proceedings of the International Conference on Machine Learning (ICML 2011)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS 2014)*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.