Restaurant Menu Generation From User Reviews

Guoxing Li
Department of Computer Science
Stanford University
Stanford, CA 94305
guoxing@stanford.edu

Tianxin Zhao
Department of Computer Science
Stanford University
Stanford, CA 94305
tianxin@stanford.edu

Abstract

User online reviews contain a great amount of non-structured information about local businesses. Yet only a small portion has been well studied. In this paper, we apply a number of different techniques in solving the named entity recognition problem in user reviews. Specifically, the named entity recognizer is used to identify and extract food entities from user reviews about local restaurant. Experimental results show that bi-directional recurrent neural networks outperform feed-forward neural networks as well as traditional CRF-based baselines.

1 Introduction

Crowd-sourced local review websites like Yelp[1] has provided people a lot of convenience on making more informative decisions on choosing local businesses. However, besides user generated review data, people are also interested in information about businesses themselves, such as menus for restaurants. By realizing most of the missing information are just buried in the large amount of user review data, we propose an approach to automatically detect and extract entities of the same type from unstructured noisy corpus. Specifically, we are interested in generating menus from user reviews on restaurants. To treat food as a category of entity, the problem further boils down to a classic Named Entity Recognition (NER) problem.

Named entity recognition has been extensively studied over the past decade. Most NER tasks involve identifying the following entity categories, PER (person), ORG (organization), LOC (location), and MISC (miscellanea). In this paper, we are interested in a specific task, identifying food entities in user review corpus, which has not been thoroughly studied before to our best knowledge. To evaluate NER systems on this specific task, we introduce a new dataset derived from Yelp Dataset Challenge [1] and Locu [2]. The task is especially challenging based on several reasons. First, the corpus is purely generated by users typing on mobile devices, which can contain typos, not properly capitalized tokens, extra punctuations, etc. As a comparison, commonly used dataset including CoNLL-03 and MUC7 are mostly generated from news corpus, which are considered to be less noisy. The same problem has been confirmed in previous work by Ritter et. al [11]. They achieved 66 F1-score on their task of named entity recognition in tweets, which share some properties with user reviews as mentioned above. Second, since one of our primary goals is to generate menus from user reviews, we are not interested in identifying food names that are too generic, e.g. salad, but more interested in specific names, e.g. Thai Chicken Salad. However, since they both represent food, it's likely that they often appear in similar context, which increases the difficulty of correctly separating them.

In seek of resolving those problems, we experiment with and analyze several different models including conditional random field (CRF), feed-forward neural network (FNN), and, bi-directional recurrent neural network (BRNN). Among those, we find that BRNN yields the best performance

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

thanks to its recurrent structure aligning well with NER's sequential nature. Our best model achieves 49.21% F1 score, which is a 27.50% increase relative to our baseline CRF system, and 2.95% increase on the multi-layer FNN system.

The rest of the paper is organized as follows. We discuss related work on named entity recognition in Section 2. Approaches including chunking scheme and training models are described in details in Section 3. In Section 4, we present our experiments and results using a newly generated dataset. We discuss results and compare performance of different models in Section 5.

2 Related Work

Our food item classification problem could boil down to a named entity recognition problem. In this section, we discuss some previous work on named entity recognition using different models.

Ratinov et al. in [5] investigated the chunk representation of named entity and showed that the BILOU encoding scheme significantly outperforms BIO encoding scheme. The paper also provides intuition on building baseline NER model based on sequential prediction problem and features about previous predictions, word context and capitalization. We built our baseline model based on these ideas. Collobert et al. in [6] presented a deep neural network architecture and use it to handle a number of NLP tasks including NER, POS tagging, etc. The deep neural network has the power to discover hidden representation of either word vectors or sentence vectors. Features could be added to the deep neural network to customize different needs. Stochastic learning is then used combined with a very big dataset to get a comparable or better result compared with traditional methods. Hammerton in [10] proposed Long Short-Term Memory(LSTM) as a recurrent neural network to model named entity recognition problem. Experiments were conducted on Reuters Corpus. They used SARDNET representation and lexical space to represent words as input, and their LSTM is one directional. The model achieved rather poor performance.

In [11], Ritter et al. presented several different techniques in solving the task of named entity recognition in tweets. The problem they were solving and ours have some commonality in that both corpus are user generated online content. We chose to investigate a generic neural network with word vectors approach instead of focusing on feature engineering in our work. Ozan and Cardie [12] applied a deep bi-directional neural network to opinion mining and achieved a new state-of-the-art performance on the task. Opinion mining is similar to NER in that they are both sequential labeling tasks on text. Our bi-directional RNN model is motivated by their work.

3 Approach

In this section, we discuss the approaches we applied to the yelp review dataset. It should be noted that classical off-the-shelf named-entity recognizers generally suffer from performance issue once applied to user generated text [11]. We therefore seek to apply neural network with word vectors to solve this problem. We first discuss chunk representation. Several different learning models are then introduced. It's worth noting that all of our models operate on sentence level contextual information, which means they don't consider context out of the sentence where the current word is at.

3.1 Chunk Representation

We use BILOU encoding scheme to represent our food entity chunk. The BILOU scheme suggests to learn classifiers that identify the **B**eginning, the **I**nside, and the **L**ast tokens of multi-token chunks as well as **U**nit-length chunks. For instance, we label food item as Honey ('B') Walnut ('I') Shrimp ('L') or Chai ('U'). We label other none-food words as 'O' class. A 5-classes classifier is used to classify each word as one of the B, I, L, U, O class.

3.2 Conditional Random Field

NER is typically viewed as a sequential prediction problem. Probabilistic models such as HMM [7] and CRF [8] have shown outstanding performance on NER. The problem can be framed as follows. Let $\mathbf{x} = (x_1, x_2, ..., x_N)$ be an input sequence of tokens, $\mathbf{y} = (y_1, y_2, ..., y_N)$ be an output sequence

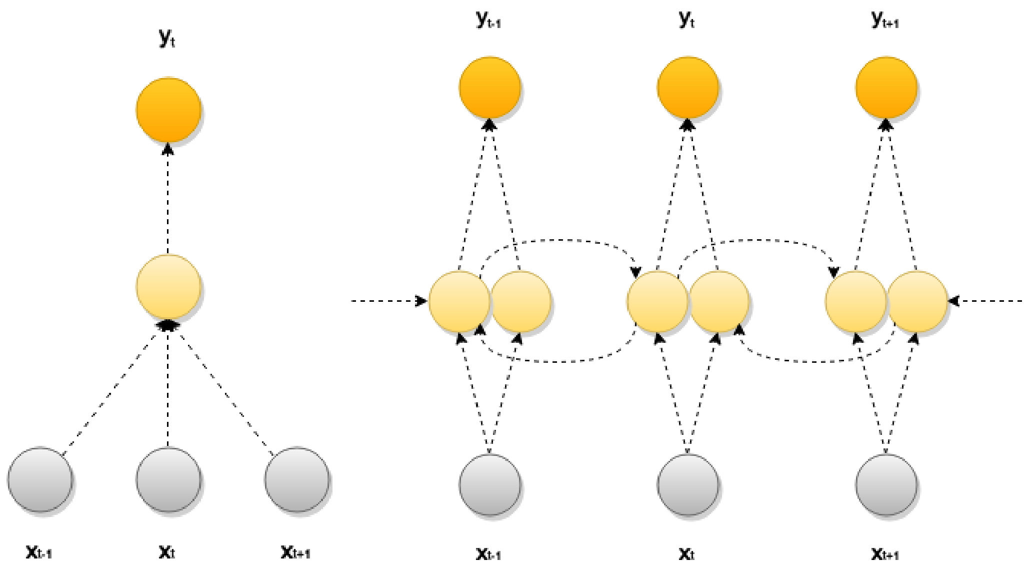

Figure 1: Demonstration of Feed-forward Neural Network of window size 3 (left), and Bi-directional Recurrent Neural Network (right).

of labels. The problem is to estimate the probabilities

$$P(y_i|x_{i-k}...x_{i+l}, y_{i-m}...y_{i-1}) \tag{1}$$

where y_i is the label for current work x_i, k, l and m are window sizes of context features. The conditional probability is estimated using the following set of features: (adapted from [9]) (1) lexicon feature of current word x_i (2) lexicon feature of previous and next word x_{i-1}, x_{i+1}. (3) capitalization pattern of current word. Note that lexicon feature means a one-hot binary vector of size n, where n is the size of the vocabulary, with each feature represents a word. For the training model, we chose to use softmax regression. For inference algorithm, it has been shown that greedy left-to-right decoding performs competitively compared to more complex and less efficient inference algorithms such as Viterbi [5]. As a result, we choose to use the greedy inference. Note that we didn't focus too much on improving the CRF model as it is not our primary model.

3.3 Feed-forward Neural Network

Feed-forward neural network is a simple type of neural network where information only flows forward from input layer, to hidden layers, and to output layer. It's suitable for named entity recognition since intuitively context around named entities are strong signals for such labeling task. The model we used is formulated as the following:

$$c_t = [L_{x_{t-w/2}}, ..., L_{x_t}, ..., L_{x_{t+w/2}}] \tag{2}$$

$$h = \tanh(Wc_t + Qs_t + b) \tag{3}$$

$$\hat{y_t} = \text{softmax}(Uh + c) \tag{4}$$

where L is the word-representation matrix, with each column L_i as the vector for a particular word i, c_t is a context window constructed by concatenating w word vectors around and including x_t, s_t is a feature vector associated with the current context window (e.g. capitalization), W, Q, b, U, and c are weight parameters, and $\hat{y}$ is the output layer that represents the probability of each label for x_t (in our case, the labels would be 'O', 'B', 'I', 'L', 'U').

Since context window at the beginning/end of the sentence is undefined, we pad special tokens '$\langle s \rangle$' at the beginning and '$\langle /s \rangle$' at the end of the sentence. Note that s_t is a flexible feature vector that incorporates information other than word vectors about the context window. For multiple contextual features, we can simply concatenate them to form a single s_t. To reduce vocabulary size, we lower-case all words in our text corpus. To restore the information lost during such transformation, we add a capitalization feature which is of dimension w. We further observe that the BILOU chunking scheme enforces labels to appear in certain orders, e.g., 'I' can only appear after 'B' or 'I'. This inspires us to add another scheme feature which consists of $w/2$ one-hot vectors with each of size 5. Another way to incorporate feature vector is to add it directly to the softmax layer. However, initial experiments achieved suboptimal performance.

Note that the neural network architecture can be easily made deeper as we add more hidden layers (h) to increase the model complexity and make it more expressive.

3.4 Bi-directional Recurrent Neural Network

Recurrent neural network (RNN) is a type of network where connections between units form a directed cycle. In other words, the hidden layer from a previous timestamp is combined with input layer to compute the hidden layer at the current timestamp. RNNs have shown great success in a range of natural language processing tasks ([mikolov lm],[sentiment socher]). It's especially useful in sequential labeling tasks such as NER due to its recurrent structure that carries information forward. However, such RNNs only have information in the past when making a decision on x_t, which is not sufficient in NER task. For instance, the context after the entity in "The[O] prime[B] rib[L] is[O] my[O] favorite[O]" gives us much more information than the context before. One way to resolve this issue is to use a bi-directional RNN (Schuster and Paliwal, 1997). We slightly modified the model, and formulate it as follows:

$$\overrightarrow{h_t} = \tanh(\overrightarrow{W} L_{x_t} + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{Q} s_t + \overrightarrow{b}) \tag{5}$$

$$\overleftarrow{h_t} = \tanh(\overleftarrow{W} L_{x_t} + \overleftarrow{V}\overleftarrow{h}_{t-1} + \overleftarrow{Q} s_t + \overleftarrow{b}) \tag{6}$$

$$\hat{y}_t = \text{softmax}(\overrightarrow{U}\overrightarrow{h}_t + \overleftarrow{U}\overleftarrow{h}_t + c) \tag{7}$$

where parameters are similar to feed-forward RNN, the ones with right arrow represents the forward pass, the ones with left arrow represents the backward pass, s_t is a feature vector that let us feed in features other than word vectors. Different from feed-foward neural network, s_t represents features only related to the current word x_t instead of the context window since such concept doesn't exist in RNN, and also RNN is capable of carrying the contextual information around. For example, for scheme feature, s_t now only contains a single one-hot vector $\text{one\_hot}(\text{argmax}_i(\hat{y}_{t-1}))$.

4 Experiments and Results

4.1 Data

Yelp Dataset Challenge contains 1.6M reviews for 61K businesses. However, they didn't provide menus for restaurants. Based on the basic information of local restaurants, we find and extract corresponding menu information from a merchant discovery service called Locu [2]. We then filter out irrelevant reviews and businesses without menu, and use Stanford Tokenizer [3] to tokenize reviews and food names on menus. We also normalize numbers to character *D*. Lastly, given a review, we label each token in an entity if it's an exact match with a food name on the same restaurant's menu without considering capitalization. The labeling is done using the BILOU encoding scheme.

We get a 69,804,176 tokens dataset after preprocessing. Since the data is highly skewed, meaning that most of the tokens are labeled as 'O' class, the model is relatively hard to fit and might miss good features. We thus took another preprocessing step to keep only sentences with two or more food entities. After this preprocessing, we have 1,585,594 tokens in total. As this dataset is still quite large and requires long time to train, we truncate it to 500,000 tokens. Finally, we divide the data as 400,000 tokens in training set, 50,000 tokens in dev set, and 50,000 tokens in test set. Each token is mapped with one class in 'B, I, L, U, O' and sentences are separated with newline. The distribution of labels after preprocessing is shown in Figure 2.

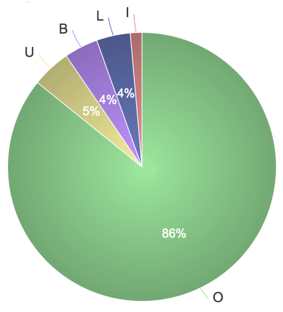

Figure 2: Label distribution (Using BILOU scheme)

4.2 Baselines

As our baseline model, we implemented a CRF using lexical features and capitalization feature mentioned in Section 3.2. A multinomial classification is needed to predict food entities with B, I, L, U classes and non-food entities with O class. Our feature vector dimension is $w * (|V| + 1)$ where w is the window size which is 3 in our case, $|V|$ is the size of the vocabulary. The vocabulary consists of 10,000 words of the most ones from the user review corpus. We choose a softmax classifier as our training model. Our baseline result is shown below on a mini dataset with 30000 training tokens and 3000 dev tokens. Larger dataset caused a memory explosion in our experiment.

4.3 Word Vectors

We use pre-trained GloVe word vectors to represents input words. We intersect our vocabulary generated from tokenizing Yelp reviews with GloVe Wikipedia [4] word vectors to form our word vectors with 71,326 words. Each word is 50 dimensions.

4.4 Evaluation Metrics

We use precision, recall and F1 score for performance evaluation. We evaluate our models on both token level F1 score and entity level F1 score. For entity level evaluation, we only count it as a true positive if it's an exact match. Note that our model may be wrong on labeling the last token in a multiple word entity (e.g. label 'L' as 'I'). Therefore, we check the label of the next token, and bound the current entity if the next label is a 'O', 'U', or 'B'. We report results in both metrics, but are more interested in entity level performance since our end goal is to identify and extract food entities from user reviews.

4.5 Training

We use a cross-entropy loss function as the objective function for all our models. In the experiment, we train three types of models including baseline CRF-based model (Baseline), feed-forward neural network (FNN), and bi-directional recurrent neural network (BRNN). We do not apply any regularization techniques as the models are not overfitting from our observation. For optimization method, we use Adagrad [13] for all of our FNN models, and stochastic gradient descent (SGD) with learning rate 0.01 for all BRNN models. We found Adagrad yields a worse performance on BRNN compared with SGD. FNN has 300 hidden units at each layer and the input layer has a fixed window size of 5. We also run a 4 hidden layer FNN with hidden dimension (1024, 1024, 256, 50). We run 15-20 epochs on the training set, and select the model with the best dev set performance. In addition to basic BRNN, we also compare BRNN with capitalization feature (BRNN+BP) and scheme feature (BRNN+SC).

4.6 Results

Quantitative Our results for different models are shown in Table 1 and Table 2. From the tables, we can see that basic BRNN performs the best on token level evaluation (51.84 F1-score). BRNN with

	Dev Set			Test Set		
	precision	recall	F1	precision	recall	F1
Baseline	31.21	23.26	26.65	28.85	20.37	23.88
FNN	**66.65**	47.30	55.02	57.44	40.16	47.10
FNN(Depth 4)	62.80	55.51	**58.59**	53.01	47.96	50.15
BRNN	62.33	**55.70**	58.53	55.90	**48.71**	**51.84**
BRNN+CP	64.41	52.45	57.30	57.80	45.48	50.43
BRNN+SC	65.17	48.97	55.11	**59.33**	41.94	48.63

Table 1: Token level evaluation on dev and test set

	Dev Set			Test Set		
	precision	recall	F1	precision	recall	F1
Baseline	27.51	20.84	23.71	25.53	18.89	21.71
FNN	61.32	41.32	49.37	53.19	36.62	43.38
FNN(Depth 4)	57.35	49.20	52.96	48.90	43.89	46.26
BRNN	57.99	**50.17**	53.80	51.54	**44.89**	47.99
BRNN+CP	60.06	46.05	52.13	53.25	41.34	46.55
BRNN+SC	**66.45**	46.44	**54.56**	**61.59**	40.98	**49.21**

Table 2: Entity level evaluation on dev and test set

scheme feature performs the best on entity level evaluation (49.21 F1-score). Note that capitalization feature makes the model performs worse. This is because user generated content can be arbitrary in capitalizing entities. Similar observation has been made in [11]. BRNN+SC performs really well on precision compared with other models (19.6% increase compared with the second best model BRNN on entity level). This is because the model learned the labeling order enforced by the scheme feature. Though it lowers recall in general.

Qualitative We generated menus for 3836 restaurants based on their reviews. We calculate the frequency of predicted food items mentioned by reviewers and construct menus. Example menus are shown in 3 and 4.

5 Discussion

Overall, the performance of all models are less than satisfactory. The best model can achieve ?? on F1 score, where the same feed-forward neural network can easily achieve over 80% F1 score on CoNLL03 dataset. We believe this is mainly due to the following reasons:

Noisy User Generated Data. User generated online data are generally very noisy. A similar observation has been identified by Ritter et al. [11] in their work on named entity recognition in tweets. Compared with traditional news-based named entity recognition datasets such as CoNLL03, user-generated data is less reliable in that, there could be typos, word abbreviations, online languages, etc.

Poor Data Labeling. The poor performance can also be ascribed to the data labeling we did was not accurate enough. We mentioned that the menu data was fetched using Locu's API, and data labeling was done by matching menu items with consecutive tokens in user reviews. However, there are a couple of problems with this approach. First, the menu data is noisy itself especially after we lower-case all tokens. For example, we've identified single word entities such as ”A”, ”Love”, ”Chicken”, ”Bacon”, etc., appear in menus. Those words are generally too generic so that our labeling approach ended up labeling them everywhere (but still in reviews about that particular restaurant) even though

RESTAURANT: Pasta China

Food Item	Frequency
Fried Rice	4
Mongolian Beef	3
Wonton Soup	3
Tso 'S Chicken	3
Sweet And Chicken	2
Shrimp Egg Rice	1
General Tso Chicken	1
Shrimp Egg Roll	1
Wonton Skins	1
Sweet And Sour Chicken	1
Roll	1
General Tso 'S Shrimp	1
Wonton Shrimp	1
Lo Mein	1
Fingers	1
Egg Fried Rice	1
Rice	1
Kung Bow Chicken	1

Table 3: Sample menu generated by a 4-hidden-layer FNN. Reviews of this restaurant are from training set

RESTAURANT: Alexi's Grill

Food Item	Frequency
Chicken Pot Pie	3
Tiger Burger	2
Black Pepper	2
Prime Rib	1
Arugula	1
White Arugula	1
Spinach Artichoke Dip	1
Jinja Ninja	1
Big Sky	1
Saison	1
Belly Bites	1
Stroganoff	1
Cheese Curds	1

Table 4: Sample menu generated by a 4-hidden-layer FNN. Reviews of this restaurant are from test set

the user didn't intend to mention the exact food name. We also notice that sometimes the food name is generic but it's actually a special kind of the generic food by reading from its descriptions. For example, there's a dish called "Mac & Cheese" but the description says it's actually "Anaheim Pepper Mac & Cheese", and the reviewers all mentioned "Anaheim Pepper Mac & Cheese" in their reviews, which made the labeling wrong. At last, food names in menus are not consistent in style. For example, there are dishes called "Alla Parmigiana Sandwich" and "Alla Parmigiana" in different menus. Though they refer to the same thing, the names in menu appear interchangeably. We used a loose entity level evaluation where we classify an entity as true positive even when one of the ending token is predicted wrong. The F1 score increases from 49.21 to 56.30.

Word Vectors Error. Word vector representation is an important part in determining the performance of our system. It's powerful in that words with similar word vectors can result in similar labelings, which makes the model able to predict on unseen words more accurately. We realize that using word vectors trained on the Wikipedia corpus is a less than optimal choice, since the corpus is drastically different from the user review dataset. This made our models perform worse. For example, the word "fav" or "fave" is used a lot as a substitute to "favorite" in user reviews. From Table 5, we can see that word vectors trained on Tweeter corpus yields much better results than on Wikipedia. However, intersecting Tweeter word vector with review vocabulary gives us far less words than Wikipedia word vector (around 30,000). A better option is to train a GloVe word vector on the entire user review corpus, and use that as the initial weights in our models. This is left as future work.

6 Conclusion

In this paper, we analyzed the performance of different models for the named entity recognition task on Yelp review dataset, including CRF-based model, feed-forward neural networks, and bi-directional recurrent neural networks.

Rank	Wikipedia	Tweeter
1	fave	fave
2	foodie	favorite
3	peepshow	favourite
4	vid	celeb
5	chupacabra	faves

Table 5: Nearest neighbors of word ’fave’ in word vectors trained on Wikipedia corpus and on Tweeter corpus

Our experiments showed that BRNNs outperformed CRF-based model as well as FNNs. Basic BRNN performed the best on token level (51.84 F1-score), and BRNN with scheme feature performed the best on entity level (49.21 F1-score). The BRNN with scheme feature yielded a significant better performance on entity level precision (19.6% increase compared with base BRNN), which is preferable since in practice we are more interested in the quality of the generated menu.

We identified that named entity recognition on user reviews is a especially challenging task, mainly due to the fact that the corpus is really noisy. We suggest further work can be focused on generating higher quality labelings. Also training a word vector representation on the user review corpus can be investigated.

References

[1] http://www.yelp.com/dataset_challenge

[2] https://locu.com

[3] http://nlp.stanford.edu/software/tokenizer.shtml

[4] http://nlp.stanford.edu/projects/glove/

[5] Ratinov, Lev, and Dan Roth. ”Design challenges and misconceptions in named entity recognition.” Proceedings of the Thirteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, 2009.

[6] Collobert, Ronan., et al. ”Natural Language Processing (Almost) from Scratch,” J. Mach. Learn. Res., 2011.

[7] Bikel, Daniel M., et al. ”Nymble: a high-performance learning name-finder.” Proceedings of the fifth conference on Applied natural language processing. Association for Computational Linguistics, 1997.

[8] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. ”Conditional random fields: Probabilistic models for segmenting and labeling sequence data.” (2001).

[9] Zhang, Tong, and David Johnson. ”A robust risk minimization based named entity recognition system.” Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, 2003.

[10] James Hammerton. ”Named Entity Recognition with Long Short-Term Memory.” Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics, 2003.

[11] Ritter, Alan, Sam Clark, and Oren Etzioni. ”Named entity recognition in tweets: an experimental study.” Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

[12] Irsoy, Ozan, and Claire Cardie. ”Opinion mining with deep recurrent neural networks.” Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.

[13] Duchi, John, Elad Hazan, and Yoram Singer. ”Adaptive subgradient methods for online learning and stochastic optimization.” The Journal of Machine Learning Research 12 (2011): 2121-2159.