MSc Dissertation

MSc in Integrated Machine Learning Systems

Student: LEI YANG

Student number: 19041732

Project Title:

Generating Recipes with Deep Neural Models

Supervisor: Ehsan Shareghi

University College London

Dept. of Electronic and Electrical Engineering

2019/2020

Abstract

The development of Artificial Intelligence has changed the concept of urban development. The concept of a smart city with a large number of smart devices such as the Internet of Things is emerging. The applications of Artificial Intelligence in smart kitchen systems could help restaurants invent more recipes or utilise their resources more effectively, or other users as a cooking assistant in order to make people's lives in a smart city more convenient.

Although there are many studies focusing on building smart applications in the field of cooking, auto-generating a recipe using Deep Neural Networks (DNN) is not well-developed. Starting from the Sequence-to-Sequence (Seq2Seq) model, we explore a series of related machine translation models, and introduce the copy mechanism-based model on the basis of the attention mechanism. We improve the Seq2Seq performance by introducing copy mechanism and pre-trained method. Besides, a variety of evaluation metrics are used to evaluate the models' performance. From the perspective of research, our work leverages Natural Language Processing (NLP) methods and Machine Translation (MT) models to tackle the task of recipe generation given a sequence of ingredients. By comparing a variety of modified models based on Seq2Seq, we find that the Seq2Seq with copy mechanism and pre-trained decoder as the best model, which can be used as the new strong baseline model for future research in this area.

The codes of our task can be found in the link.[1] The data and pre-trained embeddings are stored in the link.[2]

[1] https://github.com/yanglei-github/Recipe_Generation_Machine_Translation

[2] https://drive.google.com/drive/folders/1KUViLo9_x34lVbU_iLS8JAfy4oC2-ec-?usp=sharing

Table of Contents

1 Introduction

Since ancient times, humans have continuously invented and created recipes. These recipes are like instructions, telling others how to make delicious dishes by following instructional texts in detail. With the continuous development of human science and technology, the desire to apply existing technologies to the field of cooking is becoming stronger and stronger. In fact, a large number of researchers have begun to apply various machine learning algorithms to the field of cooking. From Natural Language Processing (NLP) perspective, this can be considered as a generation task where a recipe is generated when we input a list of ingredients. Some researchers try to use NLP method borrowed from text summarization to generate appropriate names that match existing recipe [1], while others predicted recipe through emoji and words [2]. Besides, others use Generative adversarial network (GAN) to generate delicious food images based on recipes or ingredients [3].

Nonetheless, there is little research on generating corresponding recipes from given ingredients. In particular, there is little study on applying machine learning methods to this field. And the application (how to use a trained model to help users in reality, such as a user-friendly interface or a smart device app) in this field cannot be ignored. In reality, users often look at existing ingredients in the refrigerator and then spend time thinking about how to make suitable dishes using these ingredients. However, due to the limitation of user experience in cooking, it is often impossible to apply these ingredients very efficiently, or they cannot make the type of dishes they want through these ingredients. And also, by using the recipe book, users cannot get customized cooking recipes based on exactly what ingredients they have. In this scenario, using a smart cooking assistant, users can generate recipes of certain types of dishes they want to eat based on ingredients, and then simply make the recipes through the steps produced by the assistant. Furthermore, in the future, this application can even be integrated into the refrigerator systems. By directly scanning the storage space in the refrigerator, the application can get ingredients and then display the recipes on the screen of fridges or other user interfaces after running the machine learning model.

Neural machine translation (NMT) based on Sequence-Sequence (Seq2Seq) [4] model is a successful machine translation framework compared with statistical machine translation (SMT). Few researchers have studied how to use NMT in the generating recipes task. Even though there are some studies on the NMT model in this field, most of them are trained and tuned based on private datasets, which means we cannot compare results with their models' results. Therefore, in this thesis, we not only provide high-performance NMT models, but also public our modified cleaned training dataset based on the dataset ("recipes_raw_nosource_ar") in [5].

1.1 Problem Statement

Our goal is to build a robust and reliable generating recipe model which can generate recipes of variable length (instructional texts) given a finite set of ingredients, entered by the user. Using NLP model, we hope to be able to understand and learn the inherent properties of ingredients through modelling and then generate reasonable and coherent recipes. The 'reasonable' means that most of the ingredients in the input sentence should appear in the generated recipe and the procedure of recipes should be operational. Also, the generated recipes should be readable and coherent. There are four main challenges. The first is how to deal with variable length input and output. The second is the readability and cohesiveness of the generated output. The third one is how we can make sure most of the ingredients in input appear in the recipes, and the fourth challenge is how to evaluate the quality of the model generated output. Our research and methodology would be designed and carried out in order to solve these four main challenges.

1.2 Outline of the thesis

This thesis includes five main chapters. The first chapter is the introduction, where the problem statement and outline of the thesis are provided. The second chapter is background and literature review. Sequence to Sequence is described first as the baseline model of our task. Then recipe generation and evaluation metrics of evaluating recipe generation are provided. Next, we consider optimization and pre-training methods. The third chapter introduces our models. We introduce and explain attention mechanism, copy mechanism, pre-trained encoder-decoder, and phrasal level training. The fourth chapter is the experiments and results, where preparation of dataset is described first. Then we analyze the results of training with GRU and LSTM, the results on attention and copy mechanism, the results with pre-trained embedding, the results with pre-trained encoder and decoder, and the results with phrasal level. The final chapter is conclusions, where contributions of our thesis is discussed and the potential future work is provided.

2 Background and Literature Review

Since ingredients and recipes are all composed of human language (sentences made of words), we naturally think of using NLP to solve this problem. In the past, people usually used rules-based training, which basic framework is SMT. For example, an expert sets several conditions about the prediction result and then trains the model according to these specific conditions. However, the rule-based approach relies too much on expert experience, and only generates results through a series of logical judgments. Because the experience of experts can only see the basic property of the data, it is impossible to incorporate all aspects of data. And the model we train only follows the rules which are not innovative and doesn't generatively well. Also, once the training data does not exist or does not meet the conditions set by the experts, the model output will become meaningless. As a result, the framework of NMT has been demonstrated to gradually surpass the state-of-art models of SMT [4], [6].

The Seq2Seq model is the basic model of a typical NMT model. In [7], Google proposed attention mechanism to further improve the alignment relationship between input sentence and output sentence words. The attention-based model has been demonstrated to have high-performance results in several language pairs [8]. Therefore, we introduce attention mechanism into our task because of the basic machine translation property of generating recipes. Besides, in [9], copy mechanism is introduced to guarantee more words in input sentence appear in the generated output sentence, which is an important mechanism for generating recipes based on ingredients words. In the following subsections, the architecture of Seq2Seq, the current research status in recipe generation, traditional evaluation methods for evaluating machine translation results, optimization, and pre-training are described.

2.1 Sequence to Sequence

With the development of deep learning [10], NLP can process procedural texts input through the model of recurrent neural network (RNN). In other words, generation is a very common natural language processing problem and is the key to several areas such as machine translation and summarization. RNN is an effective model for dealing with sequence-modelling problems due to its unique architecture. However, the input of RNN is multiple words, and the output is some words with the same word number. The problem is that the typical RNN model cannot deal with the task in which the number of words in the input is different from the number of words in the output, so we further introduce the Seq2Seq [4] model to deal with the problem of mismatched words.

Seq2Seq solves the problems mentioned above by introducing encoder and decoder architecture. Seq2Seq model is used in chatbot [11], machine translation [4], question answering [4], abstract text summarization [12], and text generation [12]. Among them, encoder and decoder are often designed as RNN models (GRU [13] or LSTM [14]). The input data is compressed into a context vector by the encoder, and then the context vector is passed into the decoder to get the corresponding output. Then it transfers the important information of input x (x_1, x_2, ..., x_t, where x is the input sentence instance, x_i is denoted ith word (token) in the sentences) into h_t (the hidden state at time t) through the encoder. Then through matrix multiplication and the operation of the activation function, change h_t to C (compressed vector) of a certain dimension we indicated, then input C, h_t and y_{t-1} (output at time t-1) into the decoder, predict y_t (output at current time), and then further enter c, h_{t+1} ,y_{t-1} and y_t into the next GRU or LSTM state unit until the output is <end> or <eos>.

In practical, we first use the following Eq. 1 to estimate the probability of output sequence of length T given an input sequence.

$$p\left(y_1, \dots, y_{T'} \middle| x_1, \dots, x_T\right) = \prod_{t=1}^{T'} p(y_t | x_1, \dots, x_T, y_1, \dots, y_{t-1}) \tag{1}$$

Then we can use Eq. 2 as the objective function to further train model, by maximising the probability of the target sequence T given source sequence S, where D is the whole dataset.

$$1/|D| \sum_{(T,S)\in\mathcal{S}} \log p(T|S) \tag{2}$$

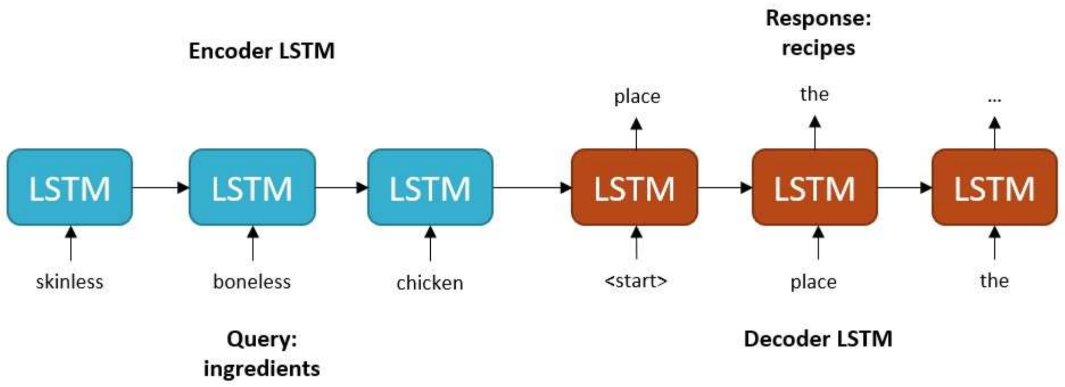


Figure 1. The basic architecture of Seq2Seq model.

In Fig. 1, we show the architecture of Seq2Seq. From Fig. 1, we can see that we input each ingredient word into one LSTM unit in the format of embedding vector in sequence in the part of encoder. Then we transfer context vector into decoder and predict each word in each LSTM unit.

2.2 Recipe Generation

Recipes generation can be regarded as a typical machine translation task (i.e., translating ingredients into recipes), which means it shares common problems in machine translation. For example, in machine translation, we can use Seq2Seq to train the English-French translation model. That is, the input is an English sentence, and the output is a French sentence with the same meaning. For our task, the input is the user-provided ingredients, and the output is recipes generated based on ingredients we provide.

It is not difficult to see from the above description about Seq2Seq that it first compresses all the input information into a context vector through Encoder. This compression operation will lose some input information (i.e, lossy compression). Also, if the input sentence is too long, some of the information entered first may not be transmitted well in the RNN model. Another problem is that classical RNN models can often only produce locally coherent language, more specifically, for long sentences, RNN often cannot generate globally coherent outputs, which means: some redundant information may be introduced, repetition may occur, readability could be compromised or some important information may not be covered.

In [12], a neural checklist model is introduced. By tracking output, the model can more efficiently improve coherence and ensure that important words appear in the output. Ingredients in input are regarded as important words. And by using this mechanism, their model can improve the probability of ingredients occurring in the output sentence. They added two additional models on the basis of the Encoder-Decoder model (Seq2Seq), one is trained to determine which items need to be generated in the subsequent output (check if there are some ingredients that were not seen in the output), and the other is trained to record input words that have already been used (for example, some certain ingredients). However, there are some problems in this method. This model forcibly uses a list to check whether certain input words are used, which means if some words have been used, it will reduce the prediction probability. However, in the real recipe, some ingredients may appear multiple times which means checking through the checklist may cause the recipe to be incoherent.

In addition, [15] proposed neural process networks that can be used to generate the next recipe sentence based on the previous recipe sentence. In neural process networks, an encoder model is used to encode sentence into a vector. Then the action selector and entity selector model use this vector to select the action and related entities. Next, the state predictor is used to update the new state. The main idea is that the model will not only learn the information of actions and entities in the input, but also the implicit information behind entities. For example, we input "add oil to the oven", through the model, we hope the model know that oil is in the oven and the temperature of the oil should be hot. This method is really useful when we generate the next sentence based on the current input sentence. However, in our task, what we have are only ingredients, which means for each prediction step, we cannot provide the corresponding previous recipe step directly due to error propagation during prediction.

2.3 Evaluation metrics

Human evaluation is not an economical and convenient method for evaluating results from machine translation [16]. In the evaluation section, commonly used automatic calculation methods are introduced to evaluate the performance of machine translation model.

2.3.1 BELU

Bilingual evaluation understudy (BLEU) [16] is often used to evaluate the quality of generated text in machine learning translation. The main idea of BLEU is to count the number of n-grams in candidate sentence appear in reference sentence regardless of the order of words. The value of it is located in the range of 0 and 1. The higher the value is, the better quality the generated text has. We can use BLEU according to different grams. BLEU-1, BLEU-2, BLEU-3 and BLEU-4 are often calculated to evaluate models' performance. The larger the number of grams, the more it can reflect the overall translation coherence of the model, and the corresponding indicators will also decrease. Besides, a source sentence can be translated into several different but reasonable target sentences. BLEU can also measure the situation where a source sentence corresponds to multiple reasonable target sentences. However, in our task, according to our dataset, we only have a source sentence with one corresponding target sentence. Therefore, we don't need this function. In this thesis, we select BLEU-1 and BLEU-4 to comprehensively measure the performance of models. Because the former can be calculated to indicate the accuracy of the predicted words, and the latter can be calculated to reflect the fluency of the predicted sentences.

The disadvantage of BLEU is that despite the introduction of brief penalty, there is still a phenomenon of higher scores for short sentences. Eq. 3 shows the calculation process of BELU, where BP is the brief penalty (the model uses BP to punish length deviations on short sentences harshly to reduce high scores in short sentences [16]), $w_n$is positive weights, p_n is the modified n-gram precisions (which is calculated by dividing the sum of clipped n-gram counts of all candidate sentences by the value of n-grams in the test corpus [16]), and N is the number of n-gram (for example, N equals 4 means that 1-gram, 2-gram, 3-gram and 4-gram are used). In Eq. 4, c refers to candidate translation length and r refers to the length of reference corpus. Eq. 4 is used to calculated BP when c⩽r. In other cases, the value of BP is always 1.

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3}$$

$$\mathrm{BP} = e^{(1-r/c)} \quad if \;\; c \leq r \tag{4}$$

2.3.2 ROUGE

Also, Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [17] is often calculated to measure the performance of translation or generation models. There are mainly four metrics in ROUGE (ROUGE-N，ROUGE-L，ROUGE-W，ROUGE-S). The difference between BLEU and ROUGE is that BLEU is calculated based on precision while ROUGE is calculated based on recall. For example, the Eq. 5 is used for calculating ROUGE-L. Where, X represents a candidate sentence, Y represents a reference sentence, LCS(X,Y) represents the length of the longest common subsequence of the candidate and the reference, m represents the length of the reference, n represents the length of the candidate, and $\beta = P_{lcs} / R_{lcs}$.

$$ROUGE - L = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{5}$$

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{6}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{7}$$

The problem of ROUGE is that it is based on word correspondence rather than semantic correspondence, which means it cannot evaluate the fluency. However, in our task, the coherence of our generated recipes is quite important. Therefore, we do not select it as the evaluation metric.

2.3.3 METEOR

Compared to BLEU, METEOR [18] introduces word alignment mechanism, which can let model consider synonyms based on the “synset” in WordNet. Besides, METEOR considers precision and recall at the same time by introducing F-score. We use Eq. 8 to calculate F-score, where P refers to precision, R refers to recall and alpha is used to adjust the weights of precision and recall.

$$F_{\text{mean}} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R} \tag{8}$$

$$\text{score} = (1 - Pen) \cdot F_{\text{mean}} \tag{9}$$

$$Pen = \gamma \cdot frag^{\beta} \tag{10}$$

$$\text{frag} \; = ch/m \tag{11}$$

In Eq. 9, F_{mean} and Pen (penalty for a given alignment) are used to calculate METEOR score. In Eq. 10, frag means a fragmentation fraction which is calculated by dividing ch (the number of chunks) by m (the number of matches) in Eq. 11, beta is the parameter for controlling shape of penalty by being the index of frag and gamma is used as the weight to tune the penalty. Because we use meteor function implemented in NLTK library [19], the same parameter values are used: which are α=0.9, β=3, γ=0.5.

2.3.4 Other metrics

In addition to the above three commonly used methods, CIDEr [20] and SPICE [21] are also commonly used. CIDEr is a combination of BLEU and vector space model. However, it has the same problem of ROUGE. It focuses on word correspondence instead of semantic correspondence. For SPICE, it uses graph-based semantic representation, which means that it mainly examines the similarity of nouns when evaluating, so it is not suitable for tasks such as machine translation.

In conclusion, we would use BLEU-1, BLEU-4 and METEOR as our evaluation metrics.

2.4 Optimization

When we train the models after we select them, we have an optimization problem at hand. Therefore, we briefly cover the optimization options here. There are many optional optimizers, such as BGD [22], SGD [22], Momentum [23], RMSprop (which is proposed by Geoff Hinton in his Coursera Class[3]), Adam [24] and Adagrad [25].

In Batch gradient descent (BGD), every time a parameter update is performed, the entire training set data needs to be used, and the training speed is very slow. Stochastic gradient descent (SGD) only needs to use one sample at a time to update parameters, which improves training efficiency. In order to further accelerate the convergence of SGD, the concept of the exponentially weighted moving average is introduced in Momentum, so that Momentum can accelerate the update of parameters in a certain parameter direction, and reduce vibration while accelerating convergence. Root mean square propagation (RMSprop) changes the gradient in the momentum update formula to the square of the gradient, which can effectively reduce the convergence efficiency of another parameter when accelerating the training of one parameter. Adam [24] combines the advantages of Momentum and RMSprop, which can be seen in the update formula. It uses the iterative method of the latter two at the same time and adds bias correction. Adagrad [25] introduces a gradient accumulation variable, which stores the sum of squared gradients before the current moment. We use Adam in our experiments.

2.4.1 Dropout and Overfitting

In deep learning, because there are lots of different parameters, once the training samples are insufficient, overfitting is easy to happen. Overfitting means that the model reaches very low training error on the training set, but generalizes poorly to the test set. Dropout [26] is used to alleviate the problem of overfitting. The idea of Dropout is to randomly ignore a certain percentage of neurons in a certain layer each time during the training process. This ratio is the dropout rate, which is specified by the user. Through the above operations, the situation that the model is too dependent on certain neurons can be reduced. In other words, each time the model is trained, it will not be completely based on the previous training experience, because each time the model can only randomly use a certain percentage of neurons for training. Dropout is demonstrated to improve the Recurrent Neural Network [27].

[3] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

2.5 Pre-training

Pre-training has a very good application effect in the Computer Vision (CV), that is, transfer learning. For image classification tasks, people usually use dataset with large number of data such as ImageNet for model pre-training. For example, ImageNet contains 1000 types of item. ImageNet-based training has already allowed parameters acquire the difference between different types of pictures. Therefore, when performing a new image classification task, the model parameters can be directly used in the new model through transfer learning, and only the last few layers of the model need to be trained in the whole process. Because pre-training parameters contain the effective information, the training time for the current model is greatly reduced.

2.5.1 Pre-trained embeddings

In the NLP field, the model converts each single word into the format of a vector and inputs it into models as input. Word2Vec [28], Glove [29], ELMo [30] and BERT [31] are commonly used to pretrain word embedding. When training a new model, we can initialize the word vectors of all input words used by the new model by using pre-trained parameters, and then continuously update the word vectors during model training to achieve a similar effect like transfer learning in CV.

ConceptNet Numberbatch [32] is a set of word vectors for word embedding trained through ConceptNet dataset, word2vec, GloVe, and OpenSubtitles 2016. Its advantage lies in not only understanding words from the context of the article but also using semi-structured, common sense knowledge from ConceptNet to understand words better [32]. Therefore, we use ConceptNet Numberbatch in our task.

2.5.2 Pre-trained encoder-decoder

Autoencoders are commonly used for pretraining model parameters or representation learning. The input of the model as the output of the model at the same time. Through training, the encoder can grasp the relevant information of the input data of the model. In reality, pre-trained encoder or decoder can be used as the implementation of auto encoder. On this basis, we can use the pre-trained encoder or decoder for further in-depth training.

3 Models

In our task, we first choose Seq2Seq as the baseline model. Because it can easily handle variable length input and output. Then we add attention mechanism [4] into the basic Seq2Seq model to further improve readability and cohesiveness of the generated output. Next, copy mechanism [9] is added into our model to make more input words appear in predicted recipes. Here we choose the copy mechanism in pointer-generator network.

To improve the model's performance, we introduce pre-trained embedding. We use pre-trained parameters to initialize the embedding layer, and continuously update these parameters during model training. After that, we still use the idea of pre-training. By using pre-trained encoder and decoder, the model is allowed to learn some information about the data before formal training. Finally, we introduce the phrasal level to allow the model to understand the data at more levels.

3.1 Attention mechanism

Traditional Seq2Seq model without attention mechanism cannot solve the alignment problem. In decoder part, each LSTM or GRU unit share the same context vector transferred from encoder part. The same context vector cannot reflect the alignment relationship between specific input word and specific output word. In [33], Google introduced attention mechanism to solve this alignment problem which has been demonstrated to have a better performance than the result of Seq2Seq model without attention mechanism.

Therefore, we use the attention mechanism to alleviate alignment problem. First, we use decoder output h_t and encoder output h_s to calculate attention weights for each LSTM or GRU unit in encoder part by using Eq. 12,

$$\alpha_{ts} = \frac{\exp\left(score(h_t, \overline{h}_s)\right)}{\sum_{s'=1}^{S} \exp\left(score(h_t, \overline{h}_{s'})\right)} \tag{12}$$

where h_t is decoder output, $\overline{h}_s$ is encoder output, $\overline{h}_{s'}$ is the specific encoder output for the s'th encoder unit, and S is the total number of encoder units.

There are many different options of score function in Eq. 12. The main purpose of using the score function is to measure the correlation between decoder output in current timestep and each encoder output. We choose Bahdanau's attention [7] as our score function, which is shown in Eq. 13, where W_1, W_2, V_a are the parameters of the model.

$$score(\boldsymbol{h_t}, \overline{\boldsymbol{h}}_s) = v_a^{\top} \tanh(W_1 h_t + W_2 \overline{h}_s) \tag{13}$$

Then, after we get attention weights, we use them to calculate unique context vector only for the current timestep of decoder by using Eq. 14,

$$\boldsymbol{c_t} = \sum_s \alpha_{ts} \overline{\boldsymbol{h}}_s \tag{14}$$

Finally, when we get the unique context vector, we concatenate it with input vector of current LSTM or GRU timestep to get attention vector by using Eq. 15,

$$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t]) \tag{15}$$

The architecture of attention mechanism is shown in Fig. 2. By storing each hidden state in encoder steps using attention mechanism, each word has its own context vector by some weights. Compared with the standard Seq2Seq, in which all output words share only one context vector, this method can better reflect the relationship between each input and output word.

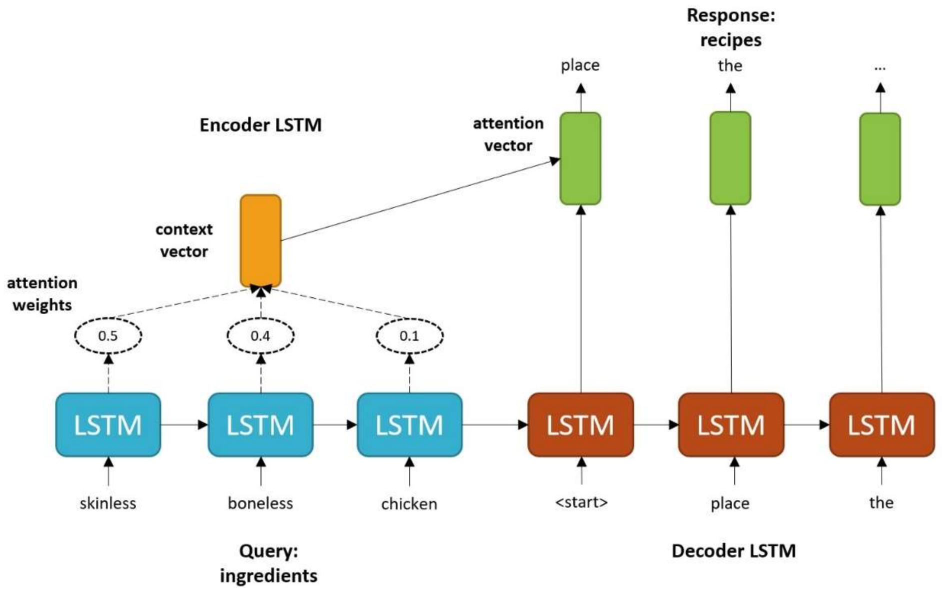


Figure 2. The architecture of Seq2Seq model with Attention mechanism.

3.2 Copy mechanism

The Seq2Seq with attention mechanism model solves the alignment problem, thereby improving the model's performance. However, it cannot further make more input ingredients words appear in generated recipes. Therefore, we introduce copy mechanism [9] to address this issue. Copy mechanism is built on attention mechanism.

The implementation of the copy mechanism needs to first calculate and save the attention weights corresponding to each input word in the attention mechanism stage. Next, find the prediction vector of each LSTM or GRU unit in the decoder part and then find the position corresponding to the input word in this vector, and add the normalized attention weights of these words to the elements at the corresponding positions in the vector, which artificially increases the probability of predicting words that have appeared in the input.

The most important point is the step of normalization. Copy mechanism has different implementation forms in different models. We chose the pointer-generator network to implement the copy mechanism. In this model, the normalized coefficient is related to context vector, the hidden state of current LSTM or GRU unit in decoder part and the input of current timestep, and the expressions of the normalized coefficient (i.e. p_{gen}) are expressed through these related parameters. Eq. 16 is used to achieve the above operations,

$$p_{\text{gen}} = \sigma\left(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}\right) \tag{16}$$

where h_t^* refers to the context vector, s_t refers to the hidden state in the current timestep, x_t refers to the input of the current timestep, and $w_{h^*}^T$, w_s^T, w_x^T and b_{ptr} are learning parameters of pointer-generator model.

Then the parameters in the normalized coefficient are trained in the training stage of our model. We can use these trained parameters to calculate different normalization coefficients for different inputs. We can further regard the normalization coefficient as a switch between the pointer network and the generator network like Eq. 17,

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + \left(1 - p_{\text{gen}}\right) \sum_{i:w_j=w} a_i^t \tag{17}$$

where a_i^t is the hidden state in encoder unit. Once p_{gen} exceeds a certain threshold, we prefer to use the generator network (this network is based on the Seq2Seq with attention mechanism model) to generate the output of the decoder at the current timestep, otherwise, the system is more inclined to use the pointer network which tends to select the appropriate word from the user input sentence as the output of the current timestep.

Fig. 3 shows the whole architecture of copy mechanism. From Fig. 3, we can see that pointer-generator network use attention weights and vector after Softmax layer to establish a direct input word and predicted output word connection.

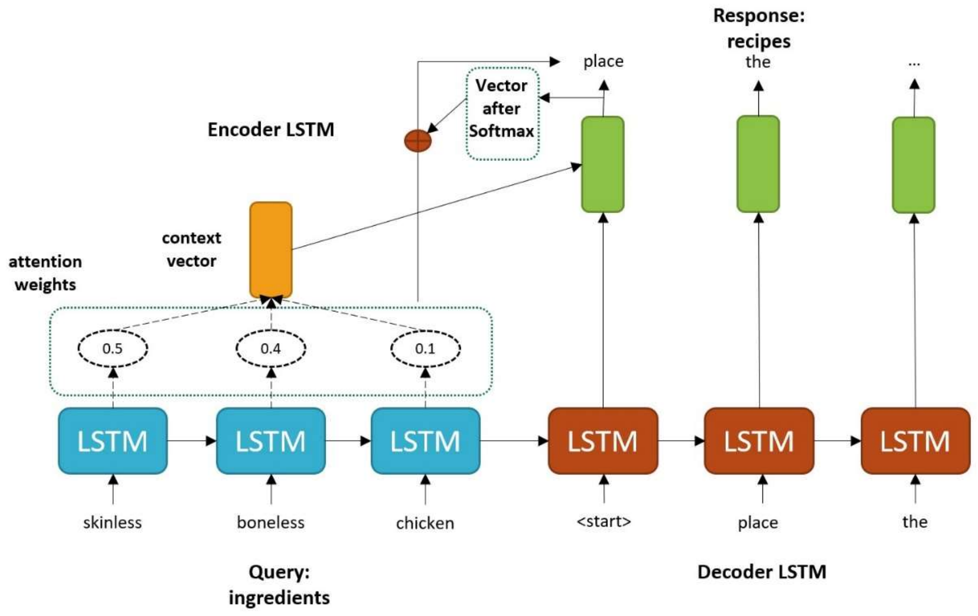


Figure 3. The architecture of Seq2Seq model with Copy mechanism.

3.3 Pre-trained encoder-decoder

We want to further improve the model's performance. Therefore, we further enhance the performance of encoder and decoder on the basis of Seq2Seq with copy mechanism. We use auto-encoding for this purpose. First, ingredients sentence is used as the input sentence and output sentence at the same time. After training 10 epochs, the decoder part is discarded, and the pre-trained encoder model is reconnected to a new decoder model. The other choice is basically the same: we keep the decoder part and use a new encoder part. The purpose of this method is to distill more information about ingredients via the pre-training process, that is, to master more prior knowledge about the input, which is equivalent to initializing the encoder with parameters containing prior knowledge.

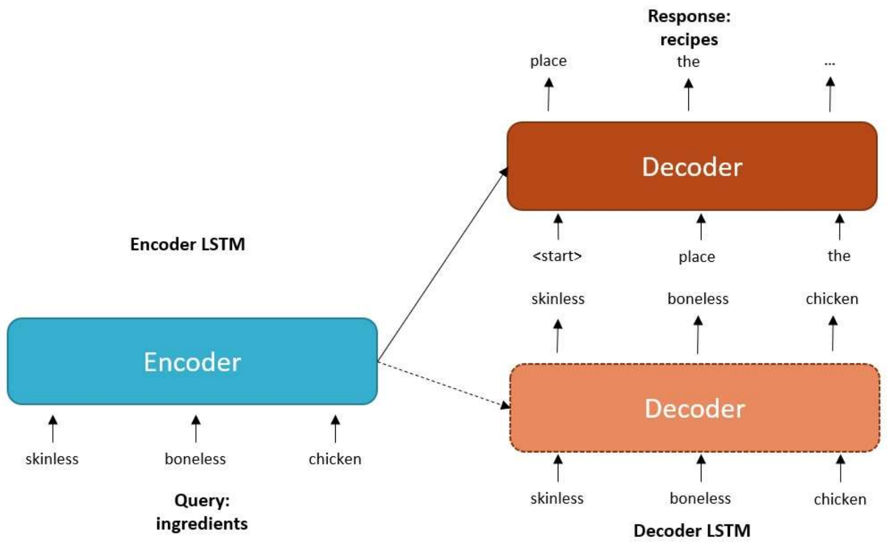


Figure 4. The architecture of Seq2Seq model with pre-trained encoder.

The architecture of pre-trained encoder and pre-trained decoder are shown in Fig. 4, 5. From Fig. 4, we can see that there are two different decoders. The first decoder is used to train with encoder, and then the pre-trained encoder reconnects to a new decoder with recipes as the output of the decoder. In Fig. 5, we pre-train the model and keep the decoder part. Then we initialize a new encoder model and connect it into pre-trained decoder.

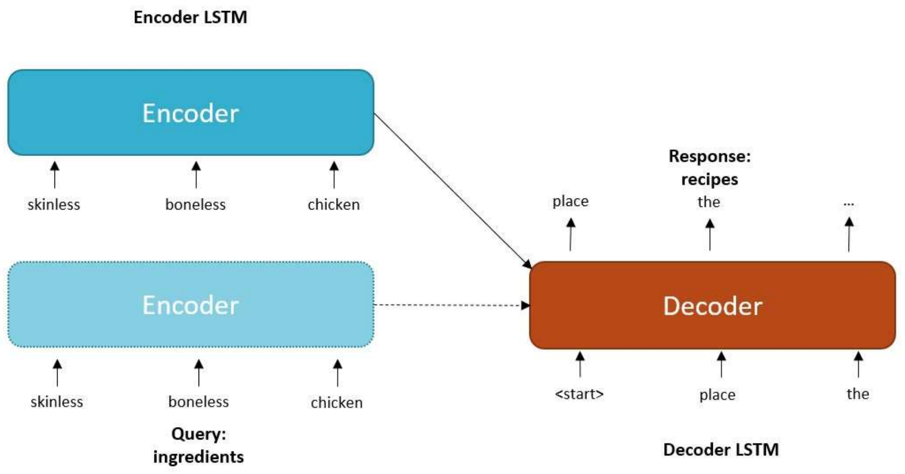


Figure 5. The architecture of Seq2Seq model with pre-trained decoder.

3.4 Phrasal level training

Most of the models with Seq2Seq with attention mechanism is trained based on the information of word level, but sometimes the information based on the word level is not enough. For example, “mashed eggs”, if we regard it as a single input unit, we allow the model to better preserve the relationship between ‘mashed’ and ‘eggs’. In [34], a phrasal level attention mechanism is proposed. However, due to the implementation of the copy mechanism, we cannot apply this method to the model based on the copy mechanism by using the same dataset. Based on the idea of auto encoder, we use pre-training to let the model learn the information on phrasal level.

With the help of the idea of a pre-trained encoder, the encoder can be trained more specifically. The key problem of the copy mechanism model is that ingredients of input sentences cannot be completely generated in the predicted recipes. By observing the input of the model, we find that the input of many ingredients has many adjectives (eg. Skinless, boneless chicken). Since our current model is based on the word level, this means that only one word can be used as the input of an LSTM unit at a time, which weakens the correlation between adjectives and nouns. The model can only rely on the encoder to learn the relationship between them. We try to use a combination of adjectives and nouns as an independent input of LSTM unit, which is equivalent to maintaining the relationship between adjectives and nouns to the greatest extent.

Through the idea of the pre-trained encoder, we first input the data divided into noun + adjective combination into the encoder. Next, we connect the encoder with a decoder and train together, and then reconnect the pre-trained encoder to a new decoder. Input the data in which nouns and adjectives are separated into the encoder and train with this new decoder. Through this process, we let the encoder learn the inner relationship between nouns and adjectives in advance, and perform normal training on the premise of retaining these relationships. It should be noted that in pre-training, we use the model based on attention mechanism rather than copy mechanism because the combination of multiple words in the encoder is regarded as an independent input vector, it is basically impossible for us to find the same vector in the output vocabulary, so the copy mechanism cannot be used. This is determined by the implementation principle of the copy mechanism. But once we have completed the pre-training and formal training, we can use the model based on the copy mechanism, because at this time our input has been switched to the normal form of a single word as the input unit.

The architecture of the model with phrasal level is shown in Fig. 6.

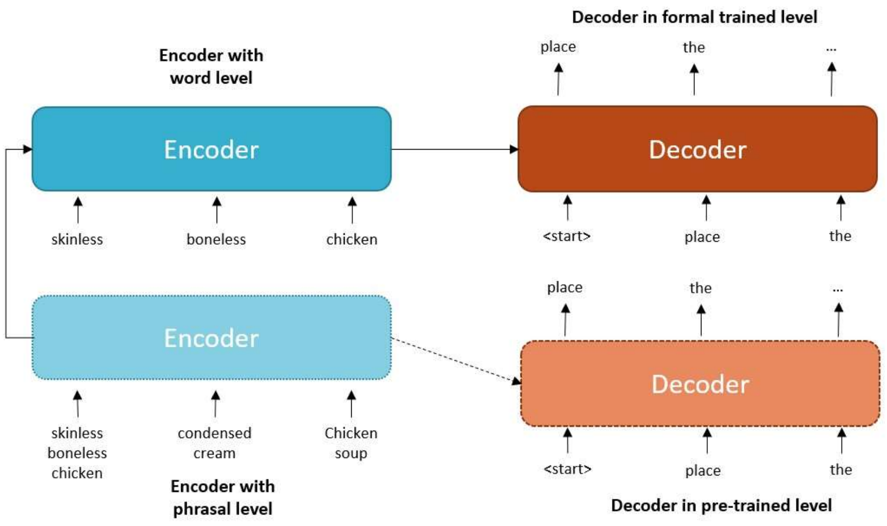


Figure 6. The architecture of Seq2Seq model with phrasal level.

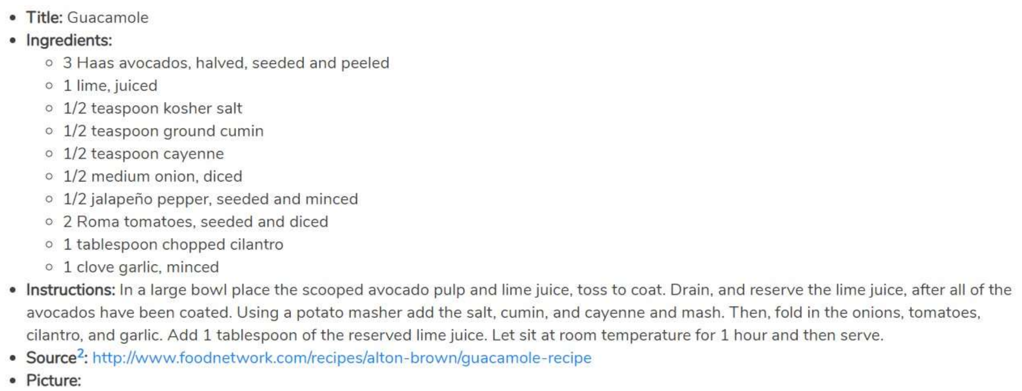
4 Experiments and Results

The process of dataset preparation is described, and then the results of training a variety of models and the analysis of results are discussed. There are few studies on recipe generation. The only model we can find to compare with our model is the neural checklist model [12], but the data set used for training this neural checklist model is too large, and our existing computing power is not enough to train our model by using the same data set. Besides, although the data set used by the model is public, the cleaned data set used in the actual training step of this model is not public. Combining the above two reasons, it is difficult for us to compare our models with neural checklist model. Nonetheless, our models are compared with strong baselines, and it can show the effectiveness of our model improvements.

For evaluating part, we choose Bilingual evaluation understudy (BLEU) [16] and Meteor to evaluate our model performance. The high values of BLEU and Meteor indicate better model performance. Besides, the coverage of the ingredients and the number of extra ingredients in the generated recipe are proposed as secondary quality metrics. The coverage of the ingredients means what percentage of input ingredients appear in the generated recipes. For this indicator, higher value means better model performance. The number of the extra ingredients means that the number of ingredients which don't appear in input but are generated in recipes. For this indicator, the small values indicate better model performance.

4.1 Preparation of dataset

We use the recipe dataset provided in [5]. The cooking examples are all acquired by scraping a variety of food and cooking websites. And the format of each example is shown in Fig 7.



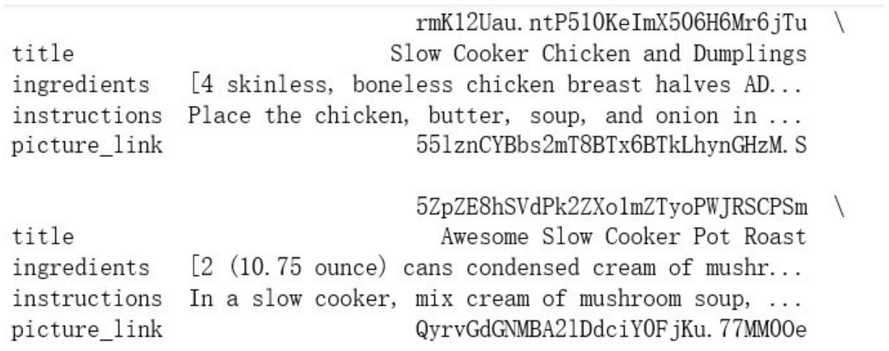
- **Title:** Guacamole
- **Ingredients:**
  - 3 Haas avocados, halved, seeded and peeled
  - 1 lime, juiced
  - 1/2 teaspoon kosher salt
  - 1/2 teaspoon ground cumin
  - 1/2 teaspoon cayenne
  - 1/2 medium onion, diced
  - 1/2 jalapeño pepper, seeded and minced
  - 2 Roma tomatoes, seeded and diced
  - 1 tablespoon chopped cilantro
  - 1 clove garlic, minced
- **Instructions:** In a large bowl place the scooped avocado pulp and lime juice, toss to coat. Drain, and reserve the lime juice, after all of the avocados have been coated. Using a potato masher add the salt, cumin, and cayenne and mash. Then, fold in the onions, tomatoes, cilantro, and garlic. Add 1 tablespoon of the reserved lime juice. Let sit at room temperature for 1 hour and then serve.
- **Source**[2]: http://www.foodnetwork.com/recipes/alton-brown/guacamole-recipe
- **Picture:**

Figure 7. The format of raw food example, which includes Title, Ingredients, Instructions, Source and Picture [5].

From Fig. 7, we can see that each raw example contains five components, which are title (name of recipes), ingredients, instructions (recipes), source (source of raw data) and the picture of food. The goal of our task is to create a model which can generate recipes based on the ingredients the users' input. Therefore, we only need the ingredients and instructions part in the raw data.

4.1.1 Data cleaning

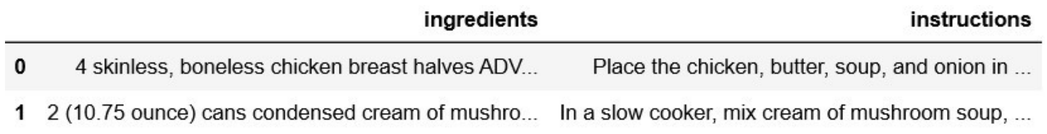
In [5], the data is in the format of json. Thus, we need to first change json format into csv format. After changing it from json into csv, we remove the title, source and picture link, and convert the data from Fig. 8 in the format which is shown in Fig. 9.



```
                               rmK12Uau.ntP510KeImX506H6Mr6jTu  \
title                              Slow Cooker Chicken and Dumplings
ingredients   [4 skinless, boneless chicken breast halves AD...
instructions  Place the chicken, butter, soup, and onion in ...
picture_link                       55lznCYBbs2mT8BTx6BTkLhynGHzM.S

                               5ZpZE8hSVdPk2ZXo1mZTyoPWJRSCPSm  \
title                                  Awesome Slow Cooker Pot Roast
ingredients   [2 (10.75 ounce) cans condensed cream of mushr...
instructions  In a slow cooker, mix cream of mushroom soup, ...
picture_link                       QyrvGdGNMBA21DdciYOFjKu.77MMOOe
```

Figure 8. The raw data [5] read from json file.



	ingredients	instructions
0	4 skinless, boneless chicken breast halves ADV...	Place the chicken, butter, soup, and onion in ...
1	2 (10.75 ounce) cans condensed cream of mushro...	In a slow cooker, mix cream of mushroom soup, ...

Figure 9. Data [5] in the format of csv.

4.1.2 Data pre-processing

The example of dataset is shown in Table 1. We can see that both input (ingredients) and output (recipes) are not that clean. For input ingredients, "ADVERTISEMENT" is the first problem. This problem is caused due to the process of scraping website, so we need to clean them. The other problem is some brackets and other notations (such as ? ! < >, and so on). Also, we need to add start token <start> and end token <end> into the input sentences and output sentences in order to train our model and let the model know when to stop. At last, we also need to change each character into lower case and add a space between a word and the punctuations (such as from "place the chicken, butter" into "place the chicken , butter"). Besides, units and quantities are also removed.

Ingredient # 1
4 skinless, boneless chicken breast halves ADVERTISEMENT 2 tablespoons butter ADVERTISEMENT 2 (10.75 ounce) cans condensed cream of chicken soup ADVERTISEMENT 1 onion, finely diced ADV ERTISEMENT 2 (10 ounce) packages refrigerated biscuit dough, torn into pieces ADVERTISEMENT A DVERTISEMENT
Recipe # 1
Place the chicken, butter, soup, and onion in a slow cooker, and fill with enough water to cover.Cov er, and cook for 5 to 6 hours on High. About 30 minutes before serving, place the torn biscuit doug h in the slow cooker. Cook until the dough is no longer raw in the center.

Table 1. The original dataset, which is acquired from [5].

After cleaning data based on the rules we set above, we can get clean data which can be used to train our model. The clean data is shown in Table 2.

Cleaned Ingredients # 1
skinless boneless chicken breast butter condensed cream chicken soup onion finely diced refrigerat ed biscuit dough torn pieces
Cleaned Recipes # 1
<start> place the chicken , butter , soup , and onion in a slow cooker , and fill with enough water to cover . cover , and cook for 5 to 6 hours on high . about 30 minutes before serving , place the torn biscuit dough in the slow cooker . cook until the dough is no longer raw in the center . <end>

Table 2. The cleaned dataset based on the original dataset [5].

4.1.3 Dataset Statistic

Full Set						
Dataset	Count of samples	Average length	std	Min of length	Max of length	Vocabulary size
Ingredients	19163	18.07	4.93	3	35	3492
recipes	19163	87.31	40.62	5	216	6728
Training Set						
Ingredients	18781	18.07	4.93	3	35	-
recipes	18781	87.23	40.63	5	216	-
Validation Set						
Ingredients	190	18.50	5.56	4	34	-
recipes	190	89.34	46.15	14	207	-
Test Set						
Ingredients	192	18.20	4.33	7	27	-
recipes	192	93.56	32.38	26	167	-

Table 3. Dataset Statistic.

We describe cleaned dataset statistic in Table 3. And our data set is divided according to the division ratio of [12]. Therefore, the whole dataset is split into training set (98%), validation set (1%) and test set (1%). Table 3 shows Average length&Standard deviation, min&max lengths and vocabulary size. We can see that mean length of the three data sets is in the range of [18,19] in ingredients and [87,93] in recipes. The average lengths of the three data sets are similar, and there is no obvious difference due to the segmentation of the data sets. For the standard deviation, although the variance of the three data sets fluctuates, it is still within an acceptable range.

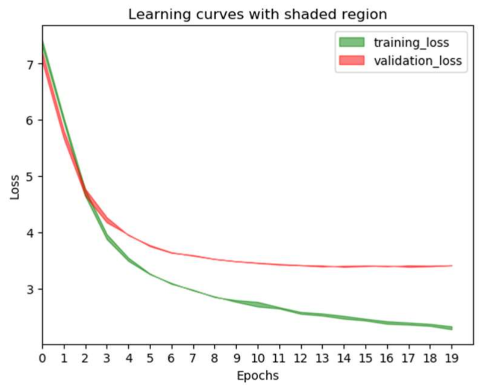
4.2 Training with GRU and LSTM

In Seq2Seq, encoder and decoder are very commonly built on GRU and LSTM. Therefore, in this section, we use both GRU and LSTM and compare the performance of them in this task. And we finally select LSTM and use it in further modified models.

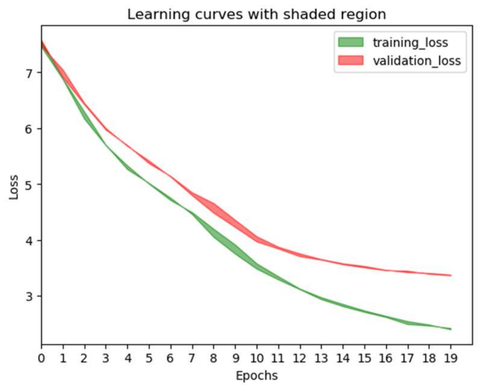
Fig. 10 shows the training and validation loss of Seq2Seq and Seq2Seq with attention models using GRU units, where the losses are computed using two separate training runs. The detailed training configurations are shown in Table 4.

Optimizer	Learning rate	Batch size	Dropout rate	Embedding size	LSTM/GRU Units	Epochs
Adam	0.01	64	0	256	256	20

Table 4. Parameters details in training phase.



(a)Loss for Seq2Seq

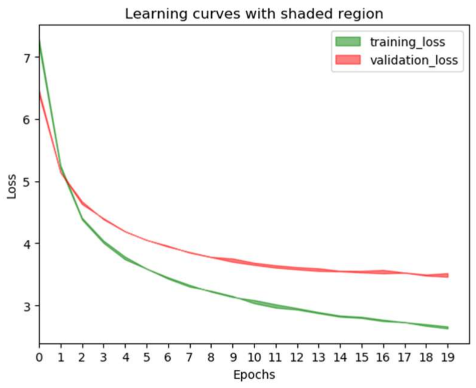


(b)Loss for Seq2Seq+Attention

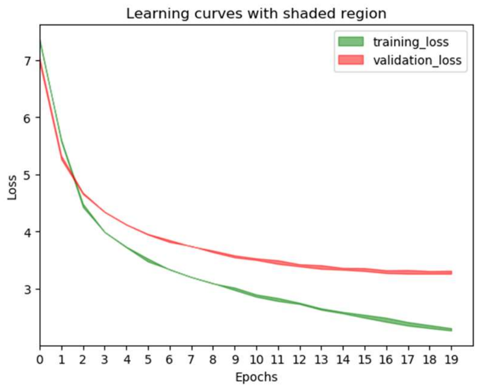
Figure 10. Training and validation loss plots for Seq2Seq, Seq2Seq+Attention with GRU, where the number is used as the unit of Epochs, and there is no unit for Loss. Shaded areas correspond to the variance of training loss and validation loss across different runs.

From Fig. 10(a), we can see that the training efficiency of Seq2Seq model without attention mechanism is very high in the first 3 epochs. The training loss and validation loss quickly drop from 7 to about 4.2, and there is no overfitting problem, but from epoch 3 to epoch 8, the decline rate of training loss and validation loss is obviously slower, and the gap between the two starts to expand gradually. Starting from epoch 9, while the training loss is slowly decreasing, the validation loss is basically stable at the value about 3.4. In Fig. 10(b), by introducing the attention mechanism, the early descending epoch continues from epoch 3 to epoch 11. Although the descending speed is not as fast as the first three epochs in Seq2Seq model without attention mechanism, it continues to be relatively efficient for more epochs in descending speed. In addition, we can notice that starting from epoch 11, although the gap between training loss and validation loss is gradually expanding, validation loss is still slowly decreasing.

Then, we keep all parameters and hyper-parameters unchanged, and replace GRU with LSTM. The training and validation loss is shown in Fig. 11.



(a)Loss for Seq2Seq



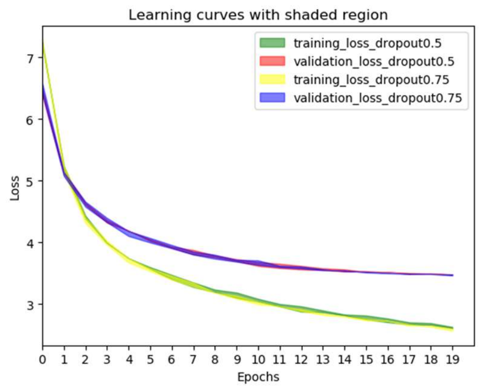
(b)Loss for Seq2Seq+Attention

Figure 11. Training and validation loss plots for Seq2Seq, Seq2Seq+Attention with LSTM, where the number is used as the unit of Epochs, and there is no unit for Loss. Shaded areas correspond to the variance of training loss and validation loss across different runs.

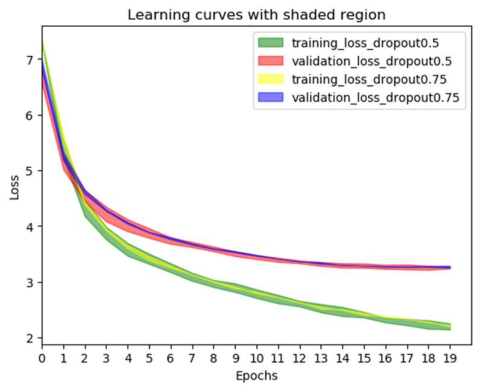
By comparing Fig. 11(a) and Fig. 11(b), we find that the introduction of the attention mechanism does not significantly improve the training efficiency of Seq2Seq model based on LSTM units, but it can slightly reduce the validation loss within the same number of epochs. Comparing Fig. 11(b) and Fig. 10(b), the introduction of the attention mechanism has a greater impact on the Seq2Seq model based on GRU units rather than LSTM units, which is reflected in two aspects. The first point is that, in GRU version, the training efficiency in the early epochs has been significantly improved, while there is no obvious improvement in LSTM version. The second point is that by observing the area of the shaded region in the two figures (shaded region responses to the stability of the model), the attention mechanism has a relatively large impact on the stability of the GRU version model. Compared Fig. 10(a) with Fig. 10(b), within epoch 7 and epoch 10, the area of the shaded region in attention mechanism is significantly larger than the model without attention mechanism, which also means that the attention mechanism has a greater impact on GRU. However, in the LSTM version model, the introduction of the attention mechanism does not cause a significant change in the area of the shaded region.

4.2.1 Dropout

From Fig. 10 and Fig. 11, we can see that whether it is GRU or LSTM, overfitting is always an existing problem. Actually, after 10 epochs, the decrease in the validation loss begins to become very negligible. Therefore, we introduce the dropout to improve this problem. Because LSTM has a better performance in the model, we choose LSTM as our standard unit both in encoder and decoder with attention mechanism. Fig. 12 shows the result of training and validation loss in Seq2Seq without/with attention mechanism in the dropout rate of 0.5 and 0.75. And the quantitative results are shown in Table 5.



(a)Loss for Seq2Seq



(b)Loss for Seq2Seq+Attention

Figure 12. Training and validation loss plots for Seq2Seq, Seq2Seq+Attention with LSTM in dropout rate of 0.5 and 0.75, where the number is used as the unit of Epochs, and there is no unit for Loss. Shaded areas correspond to the variance of training loss and validation loss across different runs.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
Seq2Seq(dropout0.5)	0.41	0.14	0.30	27	22
Seq2Seq(dropout0.75)	0.42	0.14	0.32	30	20
+Attention(dropout0.5)	0.48	0.19	0.37	43	18
+Attention(dropout0.75)	0.48	0.19	0.38	43	23

Table 5. Quantitative results on the recipe task with LSTM in dropout rate of 0.5 and 0.75 in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

In Table 5, we can compare the effects of different dropout rates on Seq2Seq and Seq2Seq with attention mechanism. In Seq2Seq model, 0.75 dropout rate has a slight increase for BLEU-1, METEOR and the average number of given items, BELU-4 is flat, and the number of extra items has a slight decrease. In Seq2Seq with attention mechanism, only METEOR has a slight improvement, and 28% of extra items have been increased, which has a serious impact on the model. Since the effect of Seq2Seq itself is not as good as that of Seq2Seq with attention mechanism, even if 0.75 dropout rate in Seq2Seq improves the model slightly, we still choose Seq2Seq with attention mechanism. But for the latter one, the introduction of

0.75 dropout rate hurts the average number of extra items in a greater degree. We choose Seq2Seq with attention mechanism in 0.5 dropout rate to compare with the model without dropout rate. We found that in addition to the slight increase in BLEU-1 and METEOR, the average number of given items and the average number of extra items were all hurt by the introduction of dropout.

In conclusion, we can suggest that using different dropout rate in our task has no obvious effect. In contrast, the introduction of dropout hurts the stability of the model training stage in the aspects of training and validation loss. Therefore, we decide not to use dropout in our main experiments.

4.3 Results on attention and copy mechanism

In this section, the evaluation of attention and copy mechanism performance is described in detail. And detailed analysis about the difference of results of attention mechanism and copy mechanism is provided. Besides, generated recipes of three sample input ingredients sentences are shown and analysed.

Through the analysis in section 4.2.1, dropout was not effective in training our model. Therefore, we introduce copy mechanism to the model without using dropout. The detailed evaluation information of models both in GRU and LSTM is shown in Table 6 and Table 7.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
Seq2Seq	0.47	0.19	0.36	32	18
+Attention	0.48	0.18	0.37	40	25
+Copy	0.44	0.18	0.37	44	18

Table 6. Quantitative results on the recipe task with GRU in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don’t appear in input but are generated in recipes.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
Seq2Seq	0.43	0.13	0.32	26	40
+Attention	0.47	0.19	0.38	44	15
+Copy	0.49	0.20	0.38	47	21

Table 7. Quantitative results on the recipe task with LSTM in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

In Table 6, we can see that Seq2Seq with attention mechanism slightly improves BLEU-1 and METEOR but reduces BLEU-4. Besides, the average number of given items has a 25% increment, which is a relatively large performance improvement. However, in the aspect of the average number of extra items, the predicted recipes introduce more extra items that do not appear in the ingredient list. Overall, we get a better model by adding attention mechanism into Seq2Seq with GRU units, at the cost increasing of extra items in generated recipes.

The improvement of the model by adding attention mechanism is reasonable. Because attention mechanism can give each GRU unit in the decoder part a unique context vector. By doing this, we can alleviate the alignment problem. By storing each hidden state in encoder steps using attention mechanism, each word has its own context vector by some weights. Compared with the situation without attention, in which all output words share one context vector, this method can better reflect the relationship between each input and output word. The most important improvement of attention mechanism is in the aspect of the average number of given items, which means attention mechanism do have an impact on solving alignment problems.

The reason for the reduction of BLEU-4 may have many reasons. The first reason we assume is that, due to the insufficient number of experiments, the average value we get may have a slightly larger variance, which may cause a deviation in the final result. Here, the reduction is 5%, which is not a large reduction value. The other assumption is that attention mechanism does hurt model's performance in BLEU-4, because the decoder might tend to predict words that have not appeared in the input sentence based on the extra alignment information introduced by the unique context vector.

More interestingly, when we change GRU unit into LSTM unit without changing any other parameters, we are delighted to see that all indicators for evaluating models have a relatively larger improvement than the GRU case, even in the aspect of the average number of extra items.

By comparing results in Table 6 and Table 7, we can demonstrate that without attention mechanism, GRU works better than LSTM in almost all items. However, once we add attention mechanism into our baseline model, LSTM has a more obvious improvement in all indicators. In addition, attention mechanism doesn't hurt the performance of the average number of extra items. In contrast, attention mechanism helps model to reduce it. The reason may come from the difference in the structure of GRU and LSTM. LSTM has a forget gate while GRU does not have a forget gate. GRU uses the update gate to calculate the value of the forget gate. Also, GRU only has the hidden state as output, while LSTM has hidden state and memory state as output. In our task, given our specific dataset, LSTM might have advantages over GRU, and the training and validation loss plot is shown in Fig. 13.

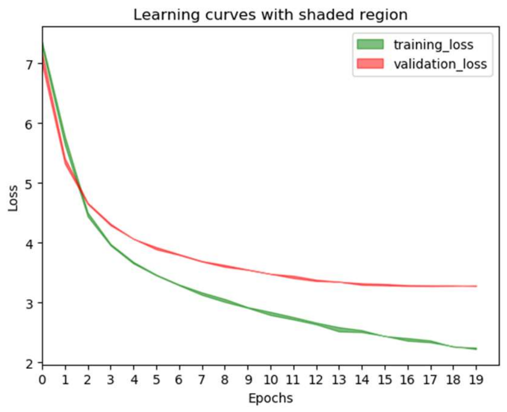


Figure 13. Training and validation loss plots for Seq2Seq+Attention+Copy with LSTM, where the number is used as the unit of Epochs, and there is no unit for Loss.

Because LSTM has a better performance in Seq2Seq with attention mechanism, we focus on the LSTM version where we introduce copy mechanism into our model. Next, we introduce copy mechanism to further make more input ingredients words appear in generated recipes. In Table 7, we can see that there is a slight improvement in the average number of given items, which shows the effect of copy mechanism.

After comparing the differences between the three models, by using hyperparameters in Table 4, we can use our trained model to predict recipes. Through these generated recipes, we can analyze the performance of the three models more intuitively. The samples are shown in Table 8, Table 9 and Table 10.

Ingredient Sample 1:	skinless boneless chicken breast butter condensed cream chicken soup onion finely diced refrigerated biscuit dough torn pieces
Gold Recipe	place the chicken , butter , soup , and onion in a slow cooker , and fill with enough water to cover . cover , and cook for 5 to 6 hours on high . about 30 minutes before serving , place the torn biscuit dough in the slow cooker . cook until the dough is no longer raw in the center .
Seq2Seq	preheat oven to 350 degrees f 175 degrees c . place chicken breasts in a 9x13 inch baking dish . pour in the soy sauce , and pour over the chicken . cover , and bake for 30 minutes . drain the chicken and discard the marinade . place the chicken breasts into the prepared baking dish . bake in the preheated oven until the chicken is no longer pink in the center , about 30 minutes . an instant read thermometer inserted into the center should read at least 165 degrees f 74 degrees c .
+Attention	preheat oven to 350 degrees f 175 degrees c . place chicken in a 9x13 inch baking dish . in a medium bowl , combine the soup , onion , and mushrooms . mix well and place in a 9x13 inch baking dish . bake at 350 degrees f 175 degrees c for 20 to 25 minutes or until chicken is cooked through and juices run clear .
+Copy	preheat oven to 350 degrees f 175 degrees c . bring a large pot of lightly salted water to a boil . add chicken and cook until tender , about 15 minutes . drain . in a large bowl , cream butter and chicken broth . add chicken and onion and mix well . pour mixture into a 9x13 inch baking dish . bake in preheated oven for 30 minutes . remove from oven and let stand for 5 minutes . remove from oven and let cool .

Table 8. Generated recipe with LSTM. Ingredient sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5]. Predicted recipes are shown respectively with Seq2Seq, Seq2Seq with attention mechanism, and Seq2Seq with attention and copy mechanism in Sample 1.[4]

Sample 2:	condensed cream mushroom soup package dry onion soup mix water pot roast
Gold Recipe	in a slow cooker , mix cream of mushroom soup , dry onion soup mix and water . place pot roast in slow cooker and coat with soup mixture . cook on high setting for 3 to 4 hours , or on low setting for 8 to 9 hours .
Seq2Seq	preheat oven to 350 degrees f 175 degrees c . in a medium bowl , mix together the mayonnaise , sour cream , and onion . mix well and pour into a 9x13 inch baking dish . bake in the preheated oven until the cheese is melted and the cheese is melted , about 30 minutes .
+Attention	preheat oven to 350 degrees f 175 degrees c . in a medium bowl , mix the soup and soup mix . pour the soup into the slow cooker . pour the marinade over the roast . cover , and bake for 1 hour in the preheated oven , or until the internal temperature of the roast has reached 145 degrees f 63 degrees c .
+Copy	preheat oven to 350 degrees f 175 degrees c . in a medium bowl , mix together the cream of mushroom soup , onion soup mix and water . place the roast in a 9x13 inch baking dish . bake in the preheated oven until the sauce is bubbly and the sauce is bubbly and the sauce is bubbly , about 45 minutes .

Table 9. Generated recipe with LSTM. Ingredient Sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5]. Predicted recipes are shown respectively with Seq2Seq, Seq2Seq with attention mechanism, and Seq2Seq with attention and copy mechanism in Sample 2.

[4] The original dataset in [5] consists of recipe data scraped from lots of different food websites.

Sample 3:	purpose flour baking soda salt butter brown sugar eggs beaten mashed overripe bananas
Gold Recipe	preheat oven to 350 degrees f 175 degrees c . lightly grease a 9x5 inch loaf pan . in a large bowl , combine flour , baking soda and salt . in a separate bowl , cream together butter and brown sugar . stir in eggs and mashed bananas until well blended . stir banana mixture into flour mixture stir just to moisten . pour batter into prepared loaf pan . bake in preheated oven for 60 to 65 minutes , until a toothpick inserted into center of the loaf comes out clean . let bread cool in pan for 10 minutes , then turn out onto a wire rack .
Seq2Seq	preheat oven to 350 degrees f 175 degrees c . grease and flour a 9 inch round cake pan . in a large bowl , cream together the butter and sugar until light and fluffy . beat in the eggs one at a time , then stir in the vanilla . sift the flour , baking soda , and salt together in a bowl . stir in the flour mixture until just incorporated . pour batter into the prepared pan . bake in the preheated oven until a toothpick inserted into the center comes out clean , about 30 minutes .
+Attention	preheat oven to 350 degrees f 175 degrees c . grease and flour a 9x13 inch pan . in a large bowl , cream together the butter and brown sugar . add the eggs and beat well . stir in the flour , bananas , and banana . stir in the banana and banana . stir in the flour until well blended . pour batter into prepared pan . bake in the preheated oven for 35 to 40 minutes , or until a toothpick inserted into the center of the cake comes out clean .
+Copy	preheat oven to 350 degrees f 175 degrees c . grease and flour a 10 inch tube pan . sift flour , baking soda , and salt together in a bowl . beat butter and brown sugar together in a bowl until smooth . beat in eggs and vanilla extract . mix in flour mixture until batter is just combined . pour batter into prepared muffin cups . bake in the preheated oven until a toothpick inserted into the center comes out clean , about 20 minutes .

Table 10. Generated recipe with LSTM. Ingredient Sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5]. Predicted recipes are shown respectively with Seq2Seq, Seq2Seq with attention mechanism, and Seq2Seq with attention and copy mechanism in Sample 3.

First, for three ingredient samples, we can find that no matter which model is used, the predicted recipe will start with "preheat oven to 350 degrees f 175 degrees c". If we collect more samples, we will find that not all generated recipes start with this sentence, but there is a higher probability that they will start with this sentence. The problem is most likely from the gold recipe in the training data set we used. There are relatively frequent gold recipes that start with this sentence, such as the gold recipe of ingredient sample 3, so this biases the model to use this sentence as the beginning sentence. This problem originating from the data set itself has no obvious impact on our exploration (that is, the exploration of the generalization of the performance of the model itself). This problem can be alleviated by using different training datasets. Therefore, we don't change the current dataset.

How can we further improve the value of the number of given items? Pre-trained embedding, pre-trained encoder and decoder and phrasal level training would be introduced to improve our current model performance. However, before that, it is necessary to figure out what is happening when using the copy mechanism, by analysing the plots of attention heatmap.

4.3.1 Attention heatmaps for Seq2Seq with attention mechanism

To further understand what are the impacts of attention mechanism and copy mechanism, attention heatmaps are used in this section, where light color corresponds to higher attention and dark color corresponds to lower attention.

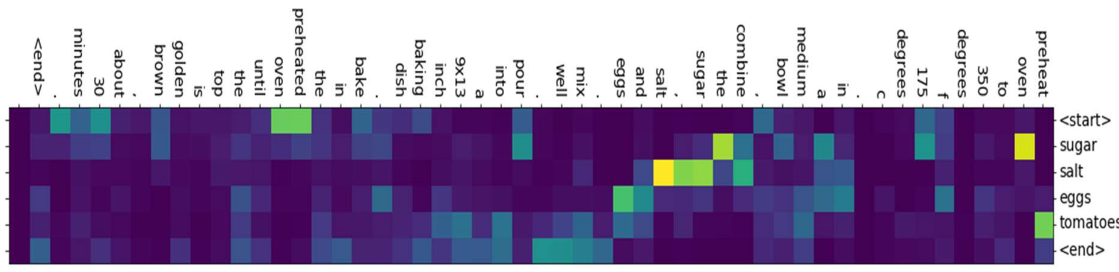


Figure 14. The attention heatmap in Seq2seq with attention in LSTM units. The text on the rows refers to the ingredients entered. The text on the columns refers to the generated recipe based on the input.

From Fig. 14, we can see that ‘salt’, ‘sugar’ and ‘eggs’, etc, occurring in the input ingredients have strong attention weights, which means attention mechanism help model to connect predicted recipe words with input ingredient words. More interesting is that ‘<start>’ in the input sentence has a strong relationship with ‘preheated’ in recipe sentence. This result indicates the generated recipe tends to start with ‘preheated’. From a realistic perspective, this situation is consistent with human perception. Because under normal circumstances, a recipe often takes heating the kitchen utensils as the first step, but this is also related to our training set. There is a slightly large proportion of data starting with ‘preheat’ in the training set, which makes the final model more inclined to use ‘preheat’ as the first word of the recipe.

However, the results also indicate that ‘<start>’ has a relatively strong attention weight with ‘175’, ’30’. The reason for this problem may come from our dataset. Because users don’t provide the model with these accurate details, which means the model doesn’t have access to this in the encoder part. Therefore, our model has to learn this information from recipes of the dataset and the relationship between input words and output words.

Next, we try to shuffle the order of input words and see the changes in generated recipes. The result is shown in Fig. 15.

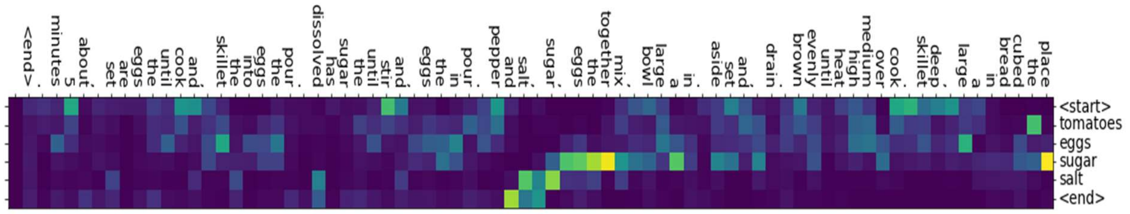


Figure 15. The attention heatmap in Seq2seq with attention in LSTM units with same ingredients but different order of Fig. 14. The text on the columns refers to the generated recipe based on the input.

By comparing the predicted recipes in Fig. 14 and Fig. 15, we can find that the model we trained is sensitive to the order of input ingredients. Inputting the same ingredients words in a different order will produce quite different results. This phenomenon is theoretically reasonable. Because the sequence model we use in the encoder part will learn the relationship between the order of each input word in the training data. Thus, can we not learn the order relationship between the input words at all? The answer is no. Because our training data contains links between each word. For example, 'mashed eggs' are two words in the input, and there is a sequence relationship between 'mashed' and 'eggs'. In future, we can further reduce the relationship between words and only retain the necessary sequence dependence between words. This point may be improved by using the Transformer [33].

Besides, what is particularly interesting is that in the generated recipe shown in Fig. 14, words such as 'baking', 'bake', and 'golden brown' appear in the generated recipe, and in the generated recipe shown in Fig. 15, 'bread', 'brown' and other words appear. From both Fig. 14 and Fig. 15, although the two recipes are different, it seems that the model will make prediction related to making bread based on 'tomatoes', 'eggs', 'sugar', 'salt'. This situation can indirectly show that the idea of introducing sequences when training the encoder will not affect the main model prediction pattern. In addition, this situation is likely to be due to a certain pattern implicit in our training data, which allows the model to tend to assume that the user is cooking bread-related dishes when the input sentence is only 'tomatoes', 'eggs', 'sugar' and 'salt', with no other ingredients. The reason why the model predicts this kind of deviation from the actual recipe topic is most likely because the number of input words is too small. However, the number of words in each input we use during training is relatively large. Therefore, we can infer that for the model trained based on our training data, the more words input, the less likely it is for the recipe predicted by the model to deviate from the topic of the recipe.

4.3.2 Attention heatmaps for Seq2Seq with copy mechanism

We use pointer-generator network to introduce copy mechanism into the current Seq2Seq with attention mechanism model to make most ingredients users provide appear in the generated recipe.

After a specific analysis of the attention map of the model with attention, we introduce the copy mechanism to visually analyze how the introduction of the copy mechanism improves the model performance, from the perspective of the attention map. The attention map of Seq2Seq with copy mechanism in LSTM units is shown in Fig. 16.

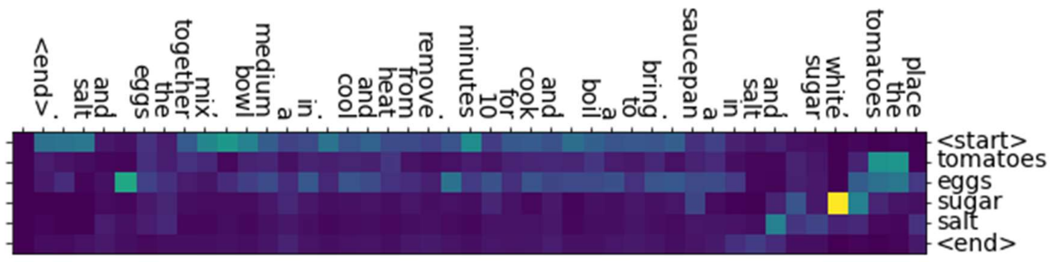


Figure 16. The attention heatmap in Seq2seq with copy mechanism in LSTM units. The text on the right refers to the sentence entered. The text on the columns refers to the generated recipe based on the input.

From Fig. 16, we can see that all the input words appear in the output recipe. However, this effect cannot be achieved in a model that only uses the attention mechanism. From Fig. 14 and Fig. 15, we can find that 'tomatoes' never appear in the predicted recipe. We can see that 'tomatoes', occurring in the input ingredients has strong attention weight. Therefore, from Fig. 16, we can intuitively see that the copy mechanism effectively improves the prediction effect of the model. Besides, more interesting is that the attention weight of 'sugar' and 'white' is higher than the value of 'sugar' and 'sugar'. We speculate that there is a certain dependence between 'white' and 'sugar', the model with copy mechanism use this relationship to further predict 'sugar' after 'white'.

4.4 Results with pre-trained embedding

In the previous model, we use randomly initialized embedding to train word embedding for the current data set. In order to further improve the model performance, pre-trained embedding is introduced to try to improve the word embedding quality, which may help improve the model effect. ConceptNet Numberbatch [32] we described in Section 2.5.1 is used as the pre-trained embedding.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items	Avg. training time per epoch(sec)
+Seq2Seq+Attention+Copy	0.49	0.20	0.38	47	21	1500
+Pre-trained embedding	0.50	0.21	0.38	43	27	5053

Table 11. Quantitative results on the recipe task in Seq2Seq+ copy mechanism and Seq2Seq+copy mechanism+pre-trained embedding in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

The results of the model with copy mechanism and the model with pre-trained embedding are shown in Table 11. It shows that pre-trained embedding has a small improvement effect on BLEU-1 and BLEU-4, but the average number of given items and the average number extra items have a tendency to deteriorate. The most important point is that the average training time per epoch of pre-trained embedding model is nearly five times that of model without pre-trained embedding, which takes a lot of time for our training, but only partially improved the final effect, so we do not introduce pre-trained embedding into our current model. Generally, pre-trained embedding can provide a better encoding of the source sentence [35]. However, in our recipe generation specific task, pre-trained embedding cannot effectively improve the performance of encoding of the source sentence. We speculate this is due to domain mismatch between the pretraining corpus and our task data.

4.5 Results with pre-trained encoder and decoder

In Table 12, the introduction of pre-trained encoder reduces the performance of the model in BLEU-1, BLEU-4, METEOR, the average number of given items, and the average number of extra items. From Fig. 17, we can see that the pre-training process of the encoder is stable and the loss declines, which shows that the encoder has indeed learned some information about the input data during the pre-training process. We speculate that the poor performance could be due to the discrepancy between the auto-encoding nature of pretraining and the training setup of the full model.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
+Copy	0.49	0.20	0.38	47	21
+Pre-trained encoder	0.45	0.17	0.34	36	23
+Pre-trained decoder	0.49	0.20	0.40	51	19

Table 12. Quantitative results on the recipe task in Seq2Seq+copy mechanism, Seq2Seq+copy mechanism+pre-trained encoder and Seq2Seq+copy mechanism+pre-trained decoder in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

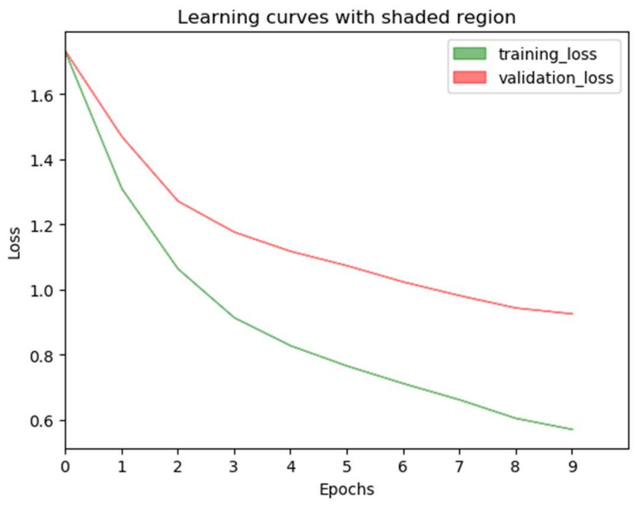


Figure 17. Training and validation loss plots for Seq2Seq+Attention+Copy with LSTM in pre-training stage, where the number is used as the unit of Epochs, and there is no unit for Loss.

Therefore, we adjust the strategy and change to pre-train the decoder. We store the knowledge learned in pre-training process into the decoder instead of the encoder. This way can ensure that in formal training, the context vector calculated by encoder will not contain misleading prior information. At the same time, the prior information in the decoder makes the decoder more inclined to preferentially select the words that appear in the input sentence when predicting the recipe, but since the target data is the recipe in the formal training, the decoder will not blindly predict the words in the input based on the prior information. The decoder will generally select words on the candidate list of recipes, but will pay more attention on words appear in this list but also appear in the input sentence.

We observe that the full model training is more effective in tuning the pretrained decoder compared with the pretrained encoder. This suggests that the distilled knowledge in the pretrained decoder (which generates the model output) increases the chance of correctly predicting ingredients at the output. An observation that was also reported in Table 12.

Ingredient Sample： *purpose flour baking soda salt butter brown sugar eggs beaten mashed overripe bananas*
Gold recipe: *preheat oven to 350 degrees f 175 degrees c . lightly grease a 9x5 inch loaf pan . in a large bowl , combine flour , baking soda and salt . in a* ***separate*** *bowl , cream together butter and brown sugar . stir in eggs and mashed bananas until well blended . stir banana mixture into flour mixture stir just to moisten . pour batter into prepared* ***loaf*** *pan . Bake in preheated oven for 60 to 65 minutes , until a toothpick inserted into center of the loaf comes out clean . let bread cool in pan for 10 minutes , then turn out onto a wire rack .*
Seq2Seq+attention+copy model: *preheat oven to 350 degrees f 175 degrees c . grease and flour a 9 inch round pan. in a large bowl , mix together flour , baking soda , and salt . cut in butter until mixture resembles coarse crumbs . stir in brown sugar and eggs . stir in bananas and mashed bananas . pour batter into prepared pan . bake in preheated oven for 60 minutes , or until a toothpick inserted into the center of the cake comes out clean .*
Seq2Seq+attention+copy+pre-trained decoder model: *preheat oven to 350 degrees f 175 degrees c . lightly grease a baking sheet . in a large bowl , combine flour , baking soda and salt . set aside . in a* ***large*** *bowl , cream together butter and brown sugar until light and fluffy . beat in eggs one at a time . stir in the banana . stir in bananas . pour batter into prepared pan . bake in preheated oven for 10 to 15 minutes , until golden brown .*

Table 13. Generated recipes sample by Seq2Seq+attention+copy model and Seq2Seq+attention+copy+pre-trained decoder model. Ingredient Sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5].

In Table 13, the text in gold recipe highlighted in red can be predicted both in Seq2Seq with copy mechanism model and Seq2Seq with pre-trained decoder. The text highlighted in blue means that only the Seq2Seq with pre-trained decoder can predict the corresponding text in the gold recipe. The text highlighted in green means that only the Seq2Seq with copy mechanism model can predict the corresponding text in the gold recipe.

From Table 13, We can intuitively find that the number of blue texts is more than that of green, which indirectly proves the results we got in Table 13. In detail, we can find that the Seq2Seq with pre-trained decoder model can predict recipes more precisely and accurately compared to the Seq2Seq with copy mechanism model. For example, Seq2Seq with pre-trained decoder model can predict that the adjective of grease is 'slightly', but the Seq2Seq with copy mechanism model cannot. In addition, the Seq2Seq with pre-trained decoder model completely predicted the blue highlighted “in a large bowl, cream together butter and brown sugar”, but this sentence was completely ignored in the Seq2Seq with copy mechanism model.

What is worth mentioning is that Seq2Seq with pre-trained decoder model predicts the recipe while adding appropriate creativity based on the property of the model itself after training.

We used purple to highlight text that can reflect the appropriate creativity. For example, “set aside”, after the current sentence is predicted, the model automatically adds this sentence (this sentence does not appear in the gold recipe) to connect the previous sentence with the next sentence to be output. The first sentence is “in a large bowl, combine flour, baking soda and salt”. The latter sentence is “in a large bowl, cream together butter and brown sugar until light and fluffy”. We can see that these two sentences are actually implemented two different operations, so the added “set aside” is very meaningful and can improve the user's reading experience. In addition, this model adds “until light and fluffy” on the basis of “in a separate bowl, cream together butter and brown sugar”. Compared with the gold recipe, we can intuitively see that the recipe with the addition of “until light and fluffy” is more detailed and professional. The reason why the model can add some connectives or adjectives appropriately is because in model training, the model learns a lot of writing patterns in training recipes, which indirectly proves the effectiveness of Seq2Seq with pre-trained decoder model training. In addition, the model also adds “one at a time after eggs”, so the text changes from “stir in eggs” to “beat in eggs one at a time”, making the operation process more detailed. In contrary, we can see the innovative phenomenon of Seq2Seq with copy mechanism model. We use yellow highlighted text to describe the innovative text automatically generated by this model. It can be found that the model directly generates an independent sentence, rather than just creatively adding some suitable adjectives or adverbs to the existing text as in the Seq2Seq with pre-trained decoder model. By comparing the text in the gold recipe, we can find that the introduced text does not have a strong correlation with the content of the gold recipe. For example, ‘crumbs’ are obviously not words in the recipe, so we can infer that the innovative sentence does not fit the context of gold recipe and generated recipe very well, and it is not appropriate to appear in the generated recipe.

4.6 Results with Phrasal level

The idea of pre-trained encoder and decoder is used to pass the phrasal level information into the encoder or decoder through pre-training method, and then the pre-trained encoder or decoder is used to perform formal training on the input data of the word level. Table 14 shows the results of the models.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
+Pre-trained decoder	0.49	0.20	0.40	51	19
+Phrasal level with pre-trained encoder	0.45	0.17	0.35	37	16
+Phrasal level with pre-trained decoder	0.49	0.19	0.38	47	15

Table 14. Quantitative results on the recipe task in Seq2Seq+copy+pre-trained decoder and Seq2Seq+copy+phrasa level with pre-trained encoder/decoder in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

In Table 14, it shows that the pre-trained encoder model with phrasal level has obvious bad effects on almost all indicators except the average number of extra items. The reason is likely to be the same as the problem of the copy mechanism model with pre-trained encoder we mentioned in the previous section. But it may also be caused by the introduction of phrasal level. The problems caused by the introduction of phrasal level exist in both pre-trained encoder and pre-trained decoder models. We would analyze them in detail in the pre-trained decoder model with phrasal level. Compared to the pre-trained encoder model with phrasal level, the pre-trained decoder version performs better, which is shown in Table 14. Therefore, we mainly compare and analyze the difference between the pre-trained decoder model with pre-trained decoder with phrasal level model. However, by comparing pre-trained decoder model with phrasal level and pre-trained decoder model, we can see that none of evaluation metrics have improved except the number of extra items.

The model learned the relationship between the vector corresponding to phrases and the output recipe in the pre-training step, but in the formal training step, due to the implementation requirements of the copy mechanism, we can only use the word level input. At this time, although the model has learned prior knowledge about the phrasal level in advance, it cannot find the corresponding vector from the word library of the recipe, so it cannot take advantage of the copy mechanism. Another reason is that the phrasal level introduces a large number of new vectors for phrases which naturally have low frequency, because every combination of words is brand new. The vocabulary size of the input data has increased from 3492 to 15173 due to the introduction of phrasal level. A large number of new vectors are difficult to effectively learn.

Ingredient Sample: *purpose flour baking soda salt butter brown sugar eggs beaten mashed overripe bananas*
Gold recipe: *preheat oven to 350 degrees f 175 degrees c . lightly grease a 9x5 inch loaf pan . in a large bowl , combine flour , baking soda and salt . in a **separate** bowl , cream together butter and brown sugar . stir in eggs and mashed bananas until well blended . stir banana mixture into flour mixture stir just to moisten . pour batter into prepared **loaf** pan . Bake in preheated oven for 60 to 65 minutes , until a toothpick inserted into center of the loaf comes out clean . let bread cool in pan for 10 minutes , then turn out onto a wire rack .*
Seq2Seq+attention+copy+phrasal-level in pre-trained decoder model: *preheat oven to 350 degrees f 175 degrees c . grease a 9x13 inch baking pan . in a large bowl , stir together the flour , baking soda , and salt . set aside . in a large bowl , cream together the butter and brown sugar until smooth . beat in the eggs one at a time , then stir in the eggs . pour the batter into the prepared pan . bake in the preheated oven until a toothpick inserted into the center of the cake comes out clean , about 45 minutes .*
Seq2Seq+attention+copy+pre-trained decoder model: *preheat oven to 350 degrees f 175 degrees c . lightly grease a baking sheet . in a large bowl , combine flour , baking soda and salt . set aside . in a large bowl , cream together butter and brown sugar until light and fluffy . beat in eggs one at a time . stir in the banana . stir in bananas . pour batter into prepared pan . bake in preheated oven for 10 to 15 minutes , until golden brown .*

Table 15. Generated recipes by Seq2Seq+attention+copy+phrasal-level in pre-trained decoder model and Seq2Seq+attention+copy+pre-trained decoder model. Ingredient Sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5].

In Table 15, the text in gold recipe highlighted in red can be predicted both by Seq2Seq with phrasal-level pre-trained decoder, and Seq2Seq with pre-trained decoder model. The text highlighted in blue means that only the Seq2Seq with pre-trained decoder model can predict the corresponding text in the gold recipe. The text highlighted in green means that only the Seq2Seq with phrasal-level pre-trained decoder model can predict the corresponding text in the gold recipe. Also, we used yellow and purple to highlight text that can reflect the creativity, respectively. By comparing the data in our table, we can see that the effect of the two models is not much different, but the introduction of “until light and fluffy” and “until golden brown” in the Seq2Seq with pre-trained decoder greatly enhance the reading experience.

We have analyzed in detail the two main reasons for the poor effect of the pre-trained decoder model with phrasal level above. The first is that the effect of the phrasal level is not fully realized, and the second is that the introduction of the phrasal level has increased the difficulty of embedding training. The second point cannot be effectively solved at present because it is determined by the property of the phrasal level itself. As long as the phrasal level is introduced, the problem of embedding must be faced. For the first point, we can try to mark the phrase in the form of phrasal level in the input and output sentences in the data pre-processing stage, and change it into an independent word. For example, the “baking soda” is identified in the input and output training data, and then changed to “baking-soda”, so that we can ensure that the phrase in the input is in the form of words in the vocabulary library of the output data, which can further improve the effect of the copy mechanism, thereby alleviating the impact of the first problem.

The quantitative result of the modified phrasal level model is shown in Table 16. And the generated recipes are shown in Table 17. The red text in Table 17 indicates the change of the phrasal level to the input and output sentences of the model. It can be seen that some words that can be regarded as phrases are combined into one word. Through the green text, we can also see the impact of input and output changes on the prediction results. For example, in the prediction result, "baking soda" becomes "baking-soda". In theory, this indicates that the phrasal level has exerted its effect, but we can see that even if the phrasal level is not used, the model can predict the baking soda relatively completely, so the advantage of phrasal level is not reflected in the current data set. In addition, we can see the blue text, which is the extra items automatically generated in the prediction result. We note that the introduction of the phrasal level increases the number of words in the word library, which increases the difficulty of the final word selection of the model. In the end, not only does the model not take advantage of the phrasal level, but the introduction of the phrasal level reduces the original prediction quality.

In conclusion, after comparing all evaluating metrics, we finally concluded that Seq2Seq model with copy mechanism in pre-trained decoder is the best model.

Model	BLEU-1	BLEU-4	METEOR	Avg. % given items	Avg. extra items
+Pre-trained decoder	0.49	0.20	0.40	51	19
+Copy	0.49	0.20	0.38	47	21
+ Modified Phrasal level	0.47	0.18	0.37	36	26

Table 16. Quantitative results on the recipe task in Seq2Seq+copy+pre-trained decoder model, Seq2Seq+copy model, and SeqSeq+modified phrasal level model in test set. Items means ingredients words. Avg. % given items means what percentage of input ingredients appear in the generated recipes. Avg. extra items means that the number of ingredients which don't appear in input but are generated in recipes.

Ingredient Sample： *purpose flour baking soda salt butter brown sugar eggs beaten mashed overripe bananas*
Gold recipe: *preheat oven to 350 degrees f 175 degrees c . lightly grease a 9x5 inch loaf pan . in a large bowl , combine flour , baking soda and salt . in a separate bowl , cream together butter and brown sugar . stir in eggs and mashed bananas until well blended . stir banana mixture into flour mixture stir just to moisten . pour batter into prepared* ***loaf*** *pan . Bake in preheated oven for 60 to 65 minutes , until a toothpick inserted into center of the loaf comes out clean . let bread cool in pan for 10 minutes , then turn out onto a wire rack .*
Seq2Seq+attention+copy model: *preheat oven to 350 degrees f 175 degrees c . grease and flour a 9 inch round pan. in a large bowl , mix together flour , baking soda , and salt . cut in butter until mixture resembles coarse crumbs . stir in brown sugar and eggs . stir in bananas and mashed bananas . pour batter into prepared pan . bake in preheated oven for 60 minutes , or until a toothpick inserted into the center of the cake comes out clean .*
Ingredients Sample purpose flour baking-soda salt butter brown-sugar eggs beaten mashed overripe bananas
Gold recipe: *preheat oven to 350 degrees f 175 degrees c . lightly grease a 9x5 inch loaf pan . in a large bowl , combine flour , baking-soda and salt . in a separate bowl , cream together butter and brown-sugar . stir in eggs and mashed bananas until well blended . stir banana mixture into flour mixture stir just to moisten . pour batter into prepared loaf pan . Bake in preheated oven for 60 to 65 minutes , until a toothpick inserted into center of the loaf comes out clean . let bread cool in pan for 10 minutes , then turn out onto a wire rack .*
Seq2Seq+attention+copy+modified phrasal-level model: *preheat oven to 350 degrees f 175 degrees c . grease a 9x13 inch baking dish . in a medium bowl , stir together the flour , baking-soda , and salt . in a separate bowl , beat together the eggs , brown-sugar , eggs , and almond-extract . stir in the flour mixture until just combined . pour the batter into the prepared pan . bake in the preheated oven until a toothpick inserted into the center comes out clean , about 45 minutes*

Table 17. Generated recipes by Seq2Seq+attention+copy model and SeqSeq+attention+copy+modified phrasal level model. Ingredient Sample and Gold Recipe is the cleaned version after our pre-processing stage based on the original dataset in [5].

5 Conclusions

In this thesis we want to build an advanced model to tackle the recipe generation task given a list of ingredients. There are four main challenges: The first is the variable length input and output. The second is the quality of the generated output. The third one is how we can make more ingredients in input appear in the recipes, and the fourth challenge is which evaluation methods should be used to evaluate the predicted output quality of the model. While solving these problems we explore a variety of deep neural models.

Through our research, we first solve the problem of variable length input and output by introducing the Seq2Seq model. After that, the attention mechanism is used to improve the readability and cohesiveness of the generated output. The introduction of the attention mechanism allows each word predicted to form a more coherent recipe through the position information brought by the attention mechanism. Based on pointer-generator network, copy mechanism is introduced to improve the performance of the model in its coverage of ingredients. The copy mechanism we introduced based on the attention mechanism can make the input words appear in the generated recipe with a higher probability so that the results predicted by our model are more practical. Copy mechanism was used in machine translation tasks, we introduce this method to the task of recipe generation.

For evaluation, we use methods from machine translation measurement metrics (BLEU and Meteor), and also introduce specific measurement metrics (the average coverage percentage of given items and the average number of extra items) for our specific task. In addition, after training the model with copy mechanism, we use attention heatmap and sampled examples to more intuitively and qualitatively evaluate the effect of the model and analyze the advantages of the copy mechanism model. Through the attention heatmap, we can observe the relationship between input and output more clearly. By comparing relationship, we can intuitively understand the role of copy mechanism. By analyzing the sampled examples, we can compare the predicted recipe with the actual recipe.

After careful analysis of the above model architectures and model results, we introduce pre-trained embedding, pre-trained encoder and decoder, and phrasal-level model to further improve the model effect. Throughout the whole process, while conducting quantitative analysis based on various indicators, we also conducted qualitative analysis based on sampled examples, analyzing the advantages and disadvantages of the models from these two aspects, and the possible reasons behind it.

All in all, we introduced the attention mechanism into the basic Seq2Seq model, and then introduced the copy mechanism into the Seq2Seq with attention mechanism model. Among a variety of improvements, the pre-trained decoder is shown to be the most effective one, and the parameters in the decoder part are retained as the initialization parameters for formal training after pre-training the model.

5.1 Future work

Based on the existing model, in future, a more customized model can be trained by changing the data set. For example, we can use data from different cuisines to train different models. In the input stage, users can not only input the ingredients but also the cuisine they want to generate. This method of splitting the data set to train multiple models can also alleviate the overfitting problems we encountered in Fig. 10 and Fig. 11 (that is, when the input ingredients are insufficient, the model judges the user's thoughts based on a priori). An effective and reliable recipe generator can make people's lives more convenient, through integration into a smart refrigerator. At the same time, in the future, more and more restaurants could use recipe generators to create creative and customized recipes for different cuisines.

In the future, we can consider introducing a transformer. Due to the particularity of the structure of the transformer itself, each word contains the information of all other words, which can effectively alleviate some of the shortcomings of Seq2Seq.

References

[1] S. Raval, “https://github.com/llSourcell/How_to_make_a_text_summarizer.”.

[2] E. Menezes, “https://docs.microsoft.com/en-us/archive/blogs/machinelearning/deep-learning-for-emojis-with-vs-code-tools-for-ai,” 2018.

[3] Rtlee9, “https://github.com/rtlee9/food-GAN,” 2017.

[4] I. Sutskever, O. Vinyals, and Q. v. Le, “Sequence to sequence learning with neural networks,” *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, 2014.

[5] R. T.Lee, “Recipe box.” [Online]. Available: https://eightportions.com/datasets/Recipes/.

[6] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, no. October, pp. 1700–1709, 2013.

[7] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[8] T. Cohn, C. D. V. Hoang, E. Vymolova, K. Yao, C. Dyer, and G. Haffari, “Incorporating structural alignment biases into an attentional neural translation model,” *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pp. 876–885, 2016.

[9] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 1073–1083, 2017.

[10] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] J. W. Wei di,Anurag Bhardwaj, “Chatbots.” [Online]. Available: https://www.oreilly.com/library/view/deep-learning-essentials/9781785880360/b71e37fb-5fd9-4094-98c8-04130d5f0771.xhtml.

[12] C. Kiddon, L. Zettlemoyer, and Y. Choi, “Globally coherent text generation with neural checklist models,” *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 329–339, 2016.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” pp. 1–9, 2014.

[14] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] A. Bosselut, O. Levy, A. Holtzman, C. Ennis, D. Fox, and Y. Choi, “Simulating action dynamics with neural process networks,” *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–15, 2018.

[16] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a Method for Automatic Evaluation of Machine Translation LK - https://rug.on.worldcat.org/oclc/204431620,” *Annual Meeting-Association for Computational Linguistics Ta - Tt* -, vol. 40, no. July, pp. 311–318, 2002.

[17] C.-Y. Lin and E. Hovy, “Automatic evaluation of summaries using N-gram co-occurrence statistics,” no. June, pp. 71–78, 2003.

[18] A. Lavie and A. Agarwal, “METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments,” *Proceedings of the Second Workshop on Statistical Machine Translation*, vol. 0, no. June, pp. 228–231, 2007.

[19] NLTK, “nltk.translate package.” [Online]. Available: http://www.nltk.org/api/nltk.translate.html#nltk.translate.meteor_score.allign_words.

[20] R. Vedantam, C. L. Zitnick, and D. Parikh, “CIDEr: Consensus-based image description evaluation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 4566–4575, 2015.

[21] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “SPICE: Semantic propositional image caption evaluation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9909 LNCS, pp. 382–398, 2016.

[22] S. Ruder, “An overview of gradient descent optimization algorithms,” pp. 1–14, 2016.

[23] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[24] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[25] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research 12, 2011.

[26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” pp. 1–18, 2012.

[27] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, “Dropout Improves Recurrent Neural Networks for Handwriting Recognition,” *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, vol. 2014-Decem, pp. 285–290, 2014.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space," 2013.

[29] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” 2014.

[30] M. E. Peters *et al.*, “Deep contextualized word representations,” *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 2227–2237, 2018.

[31] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[32] R. Speer, J. Chin, and C. Havasi, “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge,” no. Singh 2002, 2016.

[33] A. Vaswani *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[34] Y. Li, D. Xiong, and M. Zhang, “Neural machine translation with phrasal attention,” *Communications in Computer and Information Science*, vol. 787, pp. 1–8, 2017.

[35] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan, and G. Neubig, “When and why are pre-trainedword embeddings useful for neural machine translation?,” *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 2, pp. 529–535, 2018.