Capstone Project

Machine Learning Engineer Nanodegree

Ingredient Tagger

Definition

Project Overview

The goal of the project is to develop and deploy a model for extracting ingredient names, units and quantities from ingredient phrases commonly seen in cooking recipes. This can have a practical value in helping with tasks of meal planning, determining nutritional information, and other applications to the analysis of the cooking recipes.

The similar problem of tagging the parts of an ingredient phrase was solved in 2015 by the team of the New York Times using CRF in their CRF Ingredient Phrase Tagger project[1]. The dataset developed by this team was used in this project. It contains 179207 ingredient phrases which were labeled by human news assistants of the New York Times. Using this data I have tried to replicate and improve the results, applying different NLP approaches suitable for eethe problem.

Problem statement

Given a string describing a recipe ingredient string (e.g. “2 lb. of washed and peeled potatoes, cubed”), return all relevant words in the string as a quantity, units and the name of the ingredient (e.g. {“quantity”: 2, “unit”: “lb” , “ingredient”: “potatoes”}).

The tasks for solving this problem are:

- to analyze and preprocess data needed for model training
- to identify and evaluate on the ingredient dataset different models, relevant to the problem.
- to deploy the model with the best performance and implement a web interface that can be used by other services.

[1] https://github.com/nytimes/ingredient-phrase-tagger

Metrics

The metric for evaluating model performance is accuracy defined as:

$$Accuracy = \frac{number\ of\ correctly\ classified\ words\ in\ the\ dataset}{total\ number\ of\ words\ in\ the\ dataset}$$

If the dataset does not consist of significantly skewed classes this metric can be effective for model selection and tuning.

Analysis

Data exploration

Dataset for training was downloaded from the mentioned NYT project.[2]

The dataset is in csv format, here is an example of a record:

index	input	name	qty	range_end	unit	comment
0	1 1/4 cups cooked and pureed fresh butternut squash, or 1 10-ounce package frozen squash, defrosted	butternut squash	1.25	0.0	cup	cooked and pureed fresh, or 1 10-ounce package frozen squash, defrosted

The NYT project also includes a script that converts this data to a format for the CRF library used in the project. It assigns the words in a sentence their corresponding tags. Additionally, consecutive digits are clumped together with a $ sign. The resulting data is in a csv format with tab-separated columns. So the sentence above is converted to:

	0	1	2	3	4	5
0	1$1/4	I1	L20	NoCAP	NoPAREN	B-QTY

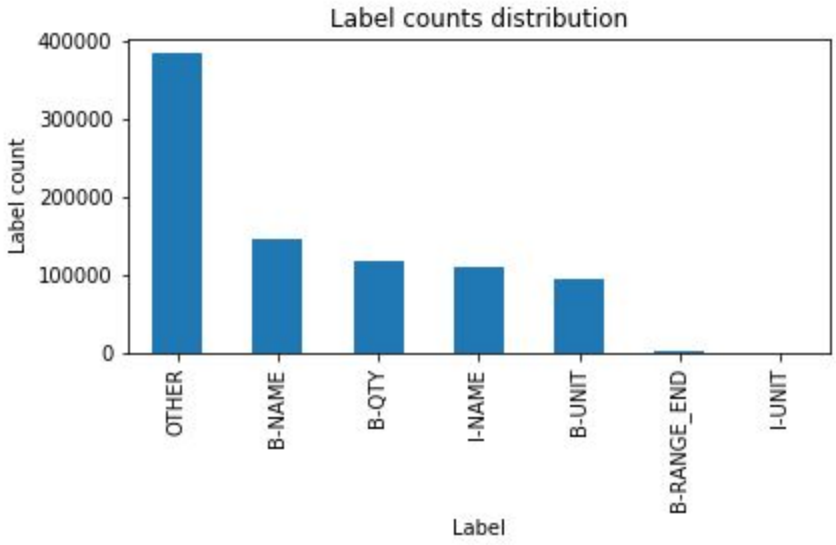
[2] https://github.com/nytimes/ingredient-phrase-tagger/blob/master/nyt-ingredients-snapshot-2015.csv.

1	cups	I2	L20	NoCAP	NoPAREN	B-UNIT
2	cooked	I3	L20	NoCAP	NoPAREN	B-COMMENT
3	and	I4	L20	NoCAP	NoPAREN	I-COMMENT
4	pureed	I5	L20	NoCAP	NoPAREN	I-COMMENT
5	fresh	I6	L20	NoCAP	NoPAREN	I-COMMENT
6	butternut	I7	L20	NoCAP	NoPAREN	B-NAME
7	squash	I8	L20	NoCAP	NoPAREN	I-NAME
8	"	I9	L20	NoCAP	NoPAREN	OTHER
9	or	I10	L20	NoCAP	NoPAREN	I-COMMENT
10	1	I11	L20	NoCAP	NoPAREN	I-COMMENT
11	10-ounce	I12	L20	NoCAP	NoPAREN	I-COMMENT
12	package	I13	L20	NoCAP	NoPAREN	I-COMMENT
13	frozen	I14	L20	NoCAP	NoPAREN	I-COMMENT
14	squash	I15	L20	NoCAP	NoPAREN	B-NAME
15	"	I16	L20	NoCAP	NoPAREN	OTHER
16	defrosted	I17	L20	NoCAP	NoPAREN	I-COMMENT

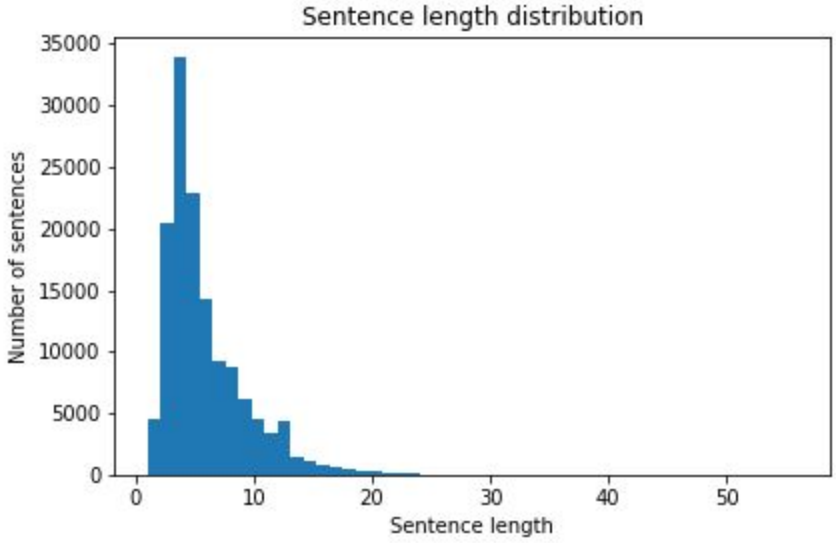
Here, the columns of interest are: 0 (words of the sentence), 1 (order of the word in the sentence) and 5 (corresponding tag). These columns are used further to generate sequences of words and corresponding labels for each ingredient phrase.

Exploratory visualization

The plot below shows the distribution of the tags after cleaning the data. It indicates that, though there are more words tagged as OTHER in ingredient phrases, the classes are not extremely skewed, in comparison to the total number of other classes, related to ingredient name, quantity and units. So accuracy can be used as a metric to evaluate and tune the models.



For batch-training and evaluating the models the distribution of ingredient phrases length can help to decide on the padding length for input sentences. The distribution on the next plot shows that 30 words is a quite safe length for padding length.



Algorithms and techniques

The problem of tagging ingredient phrases lies in the domain of Natural Language Processing and can be solved with the tools developed for the task of sequence labeling. Currently, among the most successful approaches to this task have been RNNs in combination with conditional random fields (CRF) and CNNs[3], as well as pretrained self-attention models (such as BERT).[4]

For this project I am not using CNN, since recipe ingredient phrases don't commonly include a lot of irregularities, that can be solved by incorporating character-level CNNs, and the vocabulary is limited to ingredient names, units, and food processing techniques.

The models used in this project are BiLSTM, BERT and CRF. Models were either trained from scratch (BiLSTM with Embeddings, with and without CRF layer on top) or transfer learning was used (BERT fine tuning with and without BiLSTM and CRF layers on top).

The model training was run with mini-batches of size 32 with AdamW as an optimizer for the loss function, which is negative log-likelihood for the CRF layer and cross-entropy loss for models not using CRF.

Benchmark

As a benchmark I took the model developed by the NYT team. According to the authors the accuracy of the model is 89%.[5]

There are several problems with this evaluation though:

- it is sentence-level, not token-level accuracy.
- because of a lot of noise in the data, it was computed manually on a subsample of classified data.
- I am training models on the cleaned data with a reduced number of tags that is somewhat different from the data used by the NYT project.
- it is not known what training/testing split was used by the NYT team.

So there is a need to re-evaluate the benchmark model.

To do that I re-run their algorithm on the cleaned train/test split, the same I used for training and evaluating my models and measured the token level accuracy, by comparing with the test

---

[3] See, for example, https://arxiv.org/pdf/1508.01991.pdf, https://arxiv.org/pdf/1603.01354.pdf
[4] https://arxiv.org/pdf/2010.01057v1.pdf, https://arxiv.org/pdf/2002.08902.pdf
[5] https://open.blogs.nytimes.com/2015/04/09/extracting-structured-data-from-recipes-using-conditional-random-fields/?_r=0

ground truth labels, disregarding the noise. The result was 88.7% token-level accuracy on the cleaned dataset, which is not too far from the initial value.

Methodology

Data Preprocessing

To train and evaluate different models the initial data file was randomly split into a training set of size 143269 records and testing set of size 35938 records. Then both files were processed using the data processing script from NYT CRF Ingredient Phrase Tagger project .

Data cleaning

The review of the data showed that it contained some noise in the form of the hyperlinks inside ingredient phrases as well as strings, containing tags and formatting symbols. During the cleaning stage these records were removed.

Another change that was made in original data is reducing the number of tags used for classification. For example, differriention between comment related tags (B-COMMENT, I-COMMENT) and OTHER tag was not relevant for our problem of extracting quantities, units and names of the ingredient from the ingredient phrase, so they were processed as a single tag. Also, some tagging errors were corrected during the cleaning stage.

Data preparation

After the cleaning stage, the data was prepared for model training in following steps:

- converted to word sequences with correspondent labels sequences.
- word sequences were tokenized using BertTokenizer for BERT-based models or converted to integers using the generated from training data word dictionary for LSTM-based models.
- tags were converted to integers using the tag dictionary.
- both word and tag sequences padded to the length of 30

Implementation

6 model configurations were trained and evaluated: BiLSTM, BiLSTM+CRF, BERT, BERT+BiLSTM, BERT+CRF, BERT+BiLSTM+CRF.

For training and evaluation of the models I used a Jupyter Notebook on Google Colab, to avoid the overhead of deploying the training and prediction instances on AWS.

The following hyperparameters were tuned:

- the model architecture (which combination of models performs the best on the ingredient dataset)
- learning rate.
- number of epochs to train
- embedding size for BiLSTM-based models
- hidden size for BiLSTM
- dropout for BiLSTM layer to prevent overfitting

Most hyperparameters were fine tuned manually using performance on validation dataset. To determine the number of epochs early stopping was used based on performance on the validation dataset.

Since the BERT layer had to be only fine tuned and the subsequent layers (if used) are learned from scratch, I used different learning rates, smaller for BERT and larger for subsequent layers. Moreover, training BERT with additional layers was performed in two steps, first with frozen BERT parameters and next with unfreezed parameters for all the layers. Also, a scheduler was used for gradually decreasing the learning rate as training progresses.

After evaluation on the test dataset on Google Colab, the best model (BERT as the simplest with the highest accuracy) was chosen, trained on the full dataset, and deployed on AWS as a service using SageMaker, Lambda and API Gateway.

Refinement

Using the validation set helped to determine the optimal learning rates, number of epochs and other hyperparameters for training the models.

Slight increase of the accuracy was achieved by removing the last classifying layer of the BERT model before adding BiLSTM on top. Lowercasing all the input words before processing was also tried but did not improve accuracy.

Results

Model Evaluation and Validation

The final performance of different models on the test dataset is summarized in the following table:

Model configuration	Accuracy on test dataset
BiLSTM	89.6
BiLSTM + CRF	89.3
BERT	90.1
BERT + BiLSTM	89.9
BERT + CRF	89.6
BERT + BiLSTM + CRF	89.8
Baseline	88.7

Justification

According to the results all the models showed not drastic, but slightly higher accuracy than the baseline, with the best results shown by BERT-based models. Adding BiLSTM and/or CRF layers on top of BERT did not increase the performance of the model.

During the error analysis stage we found out that around 60% of 100 randomly chosen classification errors were actually data errors in the ground truth labels, and the model classified them correctly. A large number of other labels diverging from the ground truth were due to data ambiguity. For example, some labelers tagged the whole phrase “Extra-virgin olive oil” as an ingredient name, while others tagged only “olive oil” as an ingredient name. The same was mentioned by the authors of the NYT project.[6]

It shows that it is quite difficult to achieve very high classification on this dataset. The further improvement can be achieved by data cleaning and disambiguation.

---

6 https://open.blogs.nytimes.com/2015/04/09/extracting-structured-data-from-recipes-using-conditional-random-fields

Conclusion

Reflection

For solving the problem the following steps were performed:

- identified dataset for training the model
- chose the architecture of the model
- fine tuned hyperparameters
- evaluated model on test dataset
- analyzed the classification errors
- chose and deployed the best model on AWS SageMaker
- implemented Lambda function and setup AWS Gateway API for the web access

One interesting finding was that though performance of BERT-based models showed slightly better accuracy, it was not drastically better than simple BiLISTM. I did not define the time performance in this work as a metric for model selection, since the service can be used asynchronously. But if inference time is an important parameter BiLISTM can be used.

Improvement

Given the large percentage of data error found during the manual data error analysis the biggest performance improvement probably could be achieved with data cleaning and disambiguation.

Also one can experiment with adding CNN layers to the model architecture and using different regularization techniques.

Deliverables

Application

Final deployed application is hosted here:
https://wsexg6g746.execute-api.us-west-2.amazonaws.com/prod

It an be tested with a requests like this:

curl -X POST --data '["1 large onion, sliced", "1 1/2 tablespoons finely minced garlic", "2 peeled and cubed potatoes"]' https://wsexg6g746.execute-api.us-west-2.amazonaws.com/prod

GitHub Repository

GitHub repository with the project code is at:
https://github.com/anastasia-popov/ingredient_tagger