Workshop III – Final Project

Jack 1930026143 Section: 1 & 4

Project 1: Knowledge Extraction from Unstructured Text – Extension

Intention: Relation classification (extraction). Here has some sentences, in each sentence it includes some entities and the relation between them. Now we can know what are the entities and we aim to identify the relation between them using the supervised paradigm but not OPENIE tools.

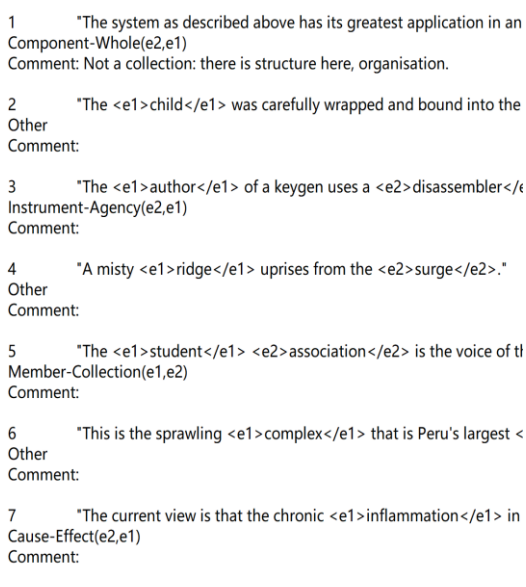
Dataset

The train and test dataset helps us label the entities with the $<\backslash e>$ structure, and each second line is relation of the entities, which means that it is the labels (y) we need. Each third line is common is almost not useful.

Basic Flow

1. Data Process

   We should extract the relation in the second line and the entities in each first line. Then we should do the embedding for each word. We use the pretrained embedding model like glove, word2vec or others to embed it.



```
1	"The system as described above has its greatest application in an
Component-Whole(e2,e1)
Comment: Not a collection: there is structure here, organisation.

2	"The <e1>child</e1> was carefully wrapped and bound into the
Other
Comment:

3	"The <e1>author</e1> of a keygen uses a <e2>disassembler</e
Instrument-Agency(e2,e1)
Comment:

4	"A misty <e1>ridge</e1> uprises from the <e2>surge</e2>."
Other
Comment:

5	"The <e1>student</e1> <e2>association</e2> is the voice of th
Member-Collection(e1,e2)
Comment:

6	"This is the sprawling <e1>complex</e1> that is Peru's largest <
Other
Comment:

7	"The current view is that the chronic <e1>inflammation</e1> in
Cause-Effect(e2,e1)
Comment:
```

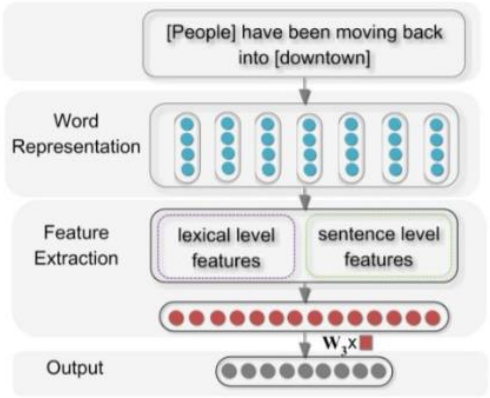
→


```
"id": "1", "relation": "Component-Whole(e2,e1)", "head": "configuration", "tail": "elements",
"id": "2", "relation": "Other", "head": "child", "tail": "cradle", "subj_start": 1, "subj_end": 1, "ol
"id": "3", "relation": "Instrument-Agency(e2,e1)", "head": "author", "tail": "disassembler", "su
"id": "4", "relation": "Other", "head": "ridge", "tail": "surge", "subj_start": 2, "subj_end": 2, "ol
"id": "5", "relation": "Member-Collection(e1,e2)", "head": "student", "tail": "association", "su
"id": "6", "relation": "Other", "head": "complex", "tail": "producer", "subj_start": 4, "subj_end"
"id": "7", "relation": "Cause-Effect(e2,e1)", "head": "inflammation", "tail": "infection", "subj_s
"id": "8", "relation": "Entity-Destination(e1,e2)", "head": "People", "tail": "downtown", "subj_
"id": "9", "relation": "Content-Container(e1,e2)", "head": "lawsonite", "tail": "platinum crucib
"id": "10", "relation": "Entity-Destination(e1,e2)", "head": "solvent", "tail": "flask", "subj_start
"id": "11", "relation": "Member-Collection(e1,e2)", "head": "essays", "tail": "volume", "subj_s
"id": "12", "relation": "Other", "head": "composer", "tail": "oblivion", "subj_start": 1, "subj_en
"id": "13", "relation": "Message-Topic(e1,e2)", "head": "citation", "tail": "reasons", "subj_star
"id": "14", "relation": "Cause-Effect(e2,e1)", "head": "burst", "tail": "pressure", "subj_start": 1,
"id": "15", "relation": "Instrument-Agency(e2,e1)", "head": "networks", "tail": "high-definitio
"id": "16", "relation": "Message-Topic(e1,e2)", "head": "call", "tail": "bill", "subj_start": 4, "sub
"id": "17", "relation": "Instrument-Agency(e2,e1)", "head": "virtuoso", "tail": "instrument", "s
"id": "18", "relation": "Product-Producer(e2,e1)", "head": "factory", "tail": "trays", "subj_start
"id": "19", "relation": "Component-Whole(e2,e1)", "head": "tree", "tail": "blossom", "subj_sta
"id": "20", "relation": "Member-Collection(e2,e1)", "head": "battalion", "tail": "grenadiers", "s
"id": "21", "relation": "Entity-Origin(e1,e2)", "head": "knowledge", "tail": "recruits", "subj_sta
"id": "22", "relation": "Member-Collection(e2,e1)", "head": "stable", "tail": "hounds", "subj_st
"id": "23", "relation": "Cause-Effect(e1,e2)", "head": "singer", "tail": "commotion", "subj_start
"id": "24", "relation": "Other", "head": "essays", "tail": "history", "subj_start": 5, "subj_end": 5,
"id": "25", "relation": "Member-Collection(e2,e1)", "head": "brace", "tail": "grouse", "subj_sta
"id": "26", "relation": "Other", "head": "dress", "tail": "blend", "subj_start": 2, "subj_end": 2, "
"id": "27", "relation": "Cause-Effect(e1,e2)", "head": "Suicide", "tail": "death", "subj_start": 0,
"id": "28", "relation": "Message-Topic(e1,e2)", "head": "article", "tail": "music", "subj_start": 1
"id": "29", "relation": "Message-Topic(e1,e2)", "head": "meeting", "tail": "rural history", "subj
"id": "30", "relation": "Component-Whole(e1,e2)", "head": "timer", "tail": "device", "subj_star
"id": "31", "relation": "Message-Topic(e2,e1)", "head": "offer", "tail": "phone call", "subj_start
"id": "32", "relation": "Cause-Effect(e2,e1)", "head": "headaches", "tail": "mold", "subj_start":
"id": "33", "relation": "Product-Producer(e1,e2)", "head": "country", "tail": "father", "subj_sta
"id": "34", "relation": "Entity-Destination(e1,e2)", "head": "stack", "tail": "carton", "subj_start
"id": "35", "relation": "Component-Whole(e1,e2)", "head": "plumbing", "tail": "manor house
"id": "36", "relation": "Entity-Origin(e1,e2)", "head": "electricity", "tail": "systems", "subj_star
"id": "37", "relation": "Other", "head": "plan", "tail": "resistance", "subj_start": 3, "subj_end":
"id": "38", "relation": "Component-Whole(e2,e1)", "head": "provinces", "tail": "counties", "su
"id": "39", "relation": "Cause-Effect(e1,e2)", "head": "stress", "tail": "divorce", "subj_start": 1,
"id": "40", "relation": "Instrument-Agency(e2,e1)", "head": "Newspapers", "tail": "service", "s
"id": "41", "relation": "Cause-Effect(e1,e2)", "head": "women", "tail": "accident", "subj_start":
"id": "42", "relation": "Content-Container(e1,e2)", "head": "transmitter", "tail": "suite", "subj_
```


2. Model Architecture

   The input to the network is a sentence, and multi-level features are extracted, where higher levels represent more abstract aspects of the input. It mainly includes the

following three components: word representation, feature extraction and output. The system does not require any complex syntactic or semantic preprocessing, and the input to the system is a sentence with two tokenized nouns. The word sequence is then converted into a word vector matrix, and next, lexical and sentence-level features are extracted separately, and then concatenated to form the final feature vector. Finally, the feature vector is fed to $softmax$ for relation classification. This is the overview of the model, after initial pretrained embedding model to represent each word at the sentence, it should perform the feature extraction in the lexical and the sentence level. Then concatenate the two levels feature vector and pass by an full connection layer to get the output, then just train the model.



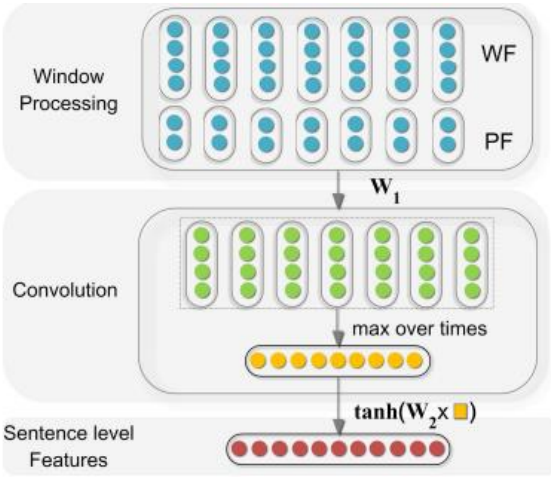
3. Feature Extraction

We know that at the data process step, we have already got the word representation (embedding) in 50-dimension. Then we should extract the feature in the text. For the lexical level, in this paper, the author selected features: word embeddings of marked nouns, the context tokens, the WordNet hypernyms, which appeared in MVRNN (Socher et al., 2012). All the features are concatenated as the lexical level feature, as follow:

Features	Remark
L1	Noun 1
L2	Noun 2
L3	Left and right tokens of noun 1
L4	Left and right tokens of noun 2
L5	WordNet hypernyms of nouns

For the sentence level which is the most important part in this model and it mirrors the convolution mentioned in the title. In this level, it includes two parts. The first part is the window processing, it divides the words representation to the WF and PF. WF is the word feature, for example, we have the sentence $(x_0, x_1, x_2, \dots, x_n)$. Then

we set the window (kernel size) to 3. Then we obtain the $WF = ([x_s, x_0, x_1], [x_0, x_1, x_2], \dots, [x_{n-1}, x_n, x_t])^T$. The we also should get the PF (Position Features). The location feature is also the innovation point of the model. The location feature describes the distance of the current word relative to the two entities nouns, then it is mapped to a low-dimensional vector $d1$ and $d2$, and then the two are directly concatenated to get the final feature. The location feature is $PF = [d1, d2]$. After that, the combination of WF and PF is $[WF, PF]^T$, which is input into the convolution module which use to extract sentence level features. First of the convolution module, it does the linear transformer for the output of the window processing $[WF, PF]^T$ and get the result Z. And then we do the max out all of Z in each row. Finally, we get the feature vector in this level.



Get the Sentence Level Feature Vector: the model uses the hyperbolic tangent function as the activation function, because the function's derivative depends only on the value of the function. And we do the nonlinear transformation $\tan(W_2 m)$ for the result m of the last step.

4. Output layer
   At this layer, the author concatenates the lexical level feature and the sentence level feature and get the final feature vector. Finally, do a linear transformation again, and the final relationship category probability is obtained after the $softmax$ classifier for classification. We should mention that we take the log-likelihood loss function.

Project 5: Traffic Sign Recognition – Extension

Intention: We know that for each type of image, we may have different accuracy for classification, so we should Find out which class has the highest classification accuracy and which has the lowest to analysis. Then in order to make the result more clearly, we should derive a confusion matrix of the traffic sign classification result on the testing set, draw a subset of the confusion matrix. Then from the confusion matrix figure, we

can know the classes that have low classification accuracy, to which classes they are mis-classified most. Finally, we can analysis the reason the result is bad.

Basic flow:

1. Firstly, we should traverse all of the labels and get the accuracy of each label. Then we can get the different accuracy for the labels by the "*predict_sign*" method in the original project. After that, we can select the best and worst labels.

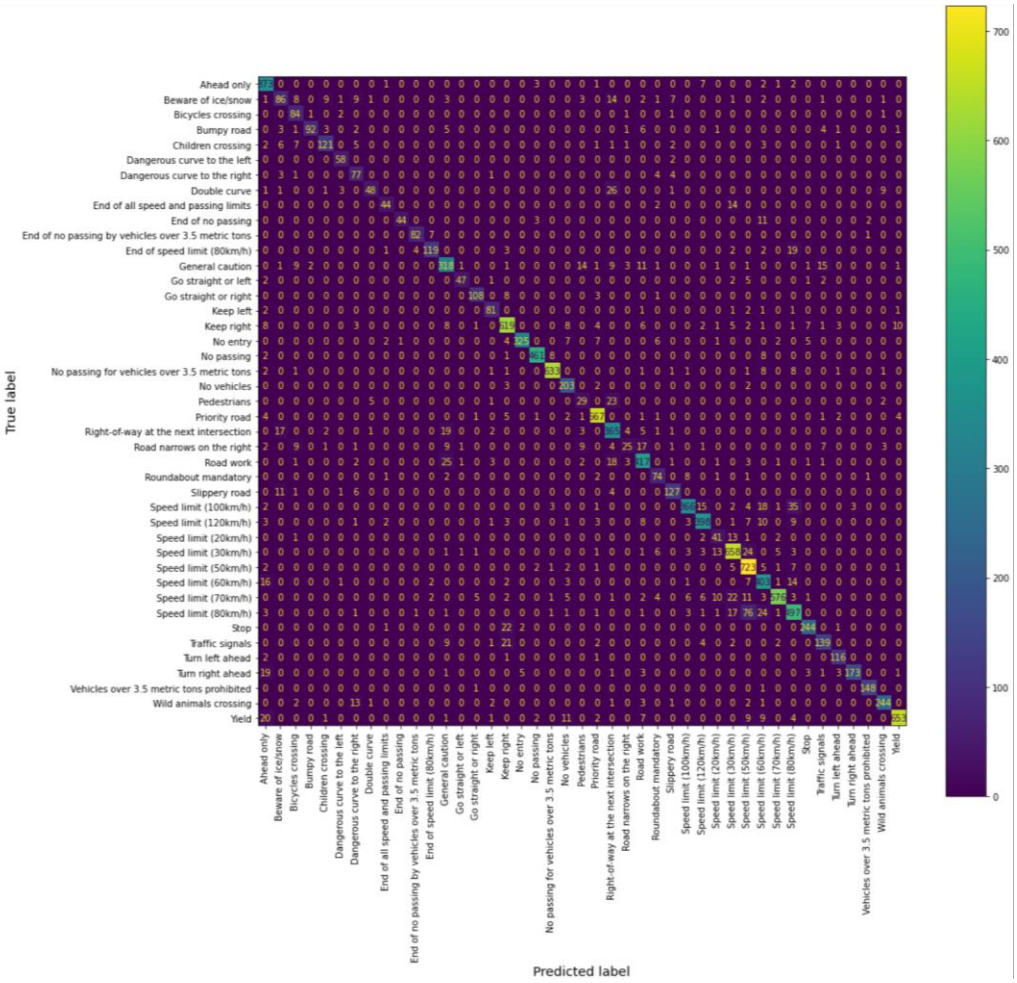
```
print('max_acc: label{}, acc = {}'.format(acc.index(max(acc)),max(acc)))
```

```
max_acc: label13, acc = 0.9875
```

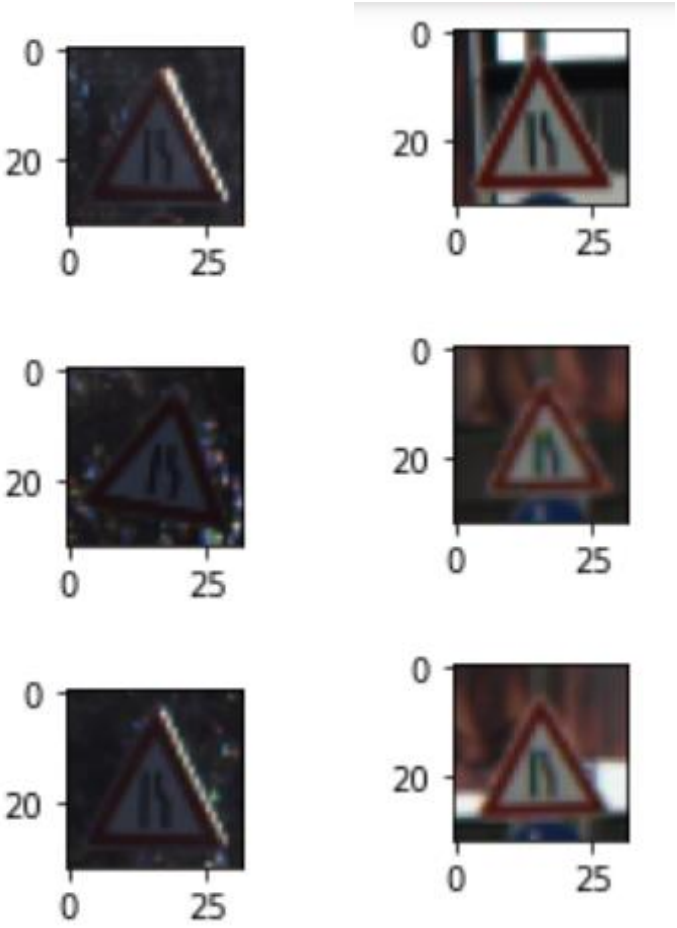
```
print('min_acc: label{}, acc = {}'.format(acc.index(min(acc)),min(acc)))
```

```
min_acc: label24, acc = 0.3333333333333333
```

2. Then, we should draw the confusion matrix figure to visualize the result. From the plot, we also can check the classification result. The darker the color, the larger the number of values in the grid, the higher the accuracy.



3. For the minimum accuracy: label 24, we can see that the color of the "*Road narrows on the right*" label is very tint and there are only 25 data predict correctly.
4. Then we can analysis the reason by observing the dataset. There are about 60 errors and we can see there are obvious gap from the correct part and the error part. In the correct part dataset, the picture is clear and bright then these errors.



5. From this result, we can know that the dataset will affect the performance of a model. So it is important to select a good dataset and use same dataset to evaluate different model.