HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

GRADUATION RESEARCH 1 REPORT

Supervisor: Do Phan Thuan

Academic year: 2020/2021

Student: Giap Thi Thuy Van – 20176905

Date: July 2021

Contents

1 Introduction

In the fourth industrial revolution, with the birth of Artificial Intelligence and Internet of Things, people would like to create a machine which can learn, think and act as human. One highlight problem is that the machine need to distinguish between objects, so that it can know that the thing is. This created one of the most important field of Machine Learning, which is called as “Classification” with the role of classifying things.
Classification in machine learning probably already has been used in various field of human life, even if people had not been aware of. If someone has a modern e-mail system, it will likely have the ability to automatically detect spam. That is, the system will analyze all incoming e-mails and mark them as either spam or not-spam. Often, the end user will be able to manually tag e-mails as spam or not, in order to improve its spam detection’s ability. This is a form of machine learning where the system is taking examples of two types of messages: spam and ham (the typical term for “non spam e-mails”) and using these examples to automatically classify incoming e-mails.
The general method of classification is to use a set of examples of each class to learn rules that can be applied to new examples. This is one of the most important machine learning modes and is the topic of this project.

2 Theory

I learned “Deep learning specialization course” on Coursera website. That course includes five sub-courses that are mentioned below.

2.1 Neural Networks and Deep Learning

There are 4 sections in this course:

- Introduction to Deep Learning
- Neural Networks Basics
- Shallow Neural Networks
- Deep Neural Networks

2.2 Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization

There are 3 sections in this course:

- Practical Aspects of Deep Learning
- Optimization Algorithms
- Hyperparameter Tuning, Batch Normalization and programming Frameworks

2.3 Structuring Machine Learning Projects

Machine learning strategy:

2.4 Convolution Neural Networks

There are 4 sections in this course:

- Foundations of Convolution Neural Networks
- Deep Convolution Models: Case Studies
- Object Detection
- Special Applications: Face recognition & Neural Style Transfer

2.5 Sequence Models

There are 4 sections in this course:

- Recurrent Neural Networks
- Natural Language Processing & Word Embedding
- Sequence Model & Attention Mechanism
- Transformer Networks

3 Practice

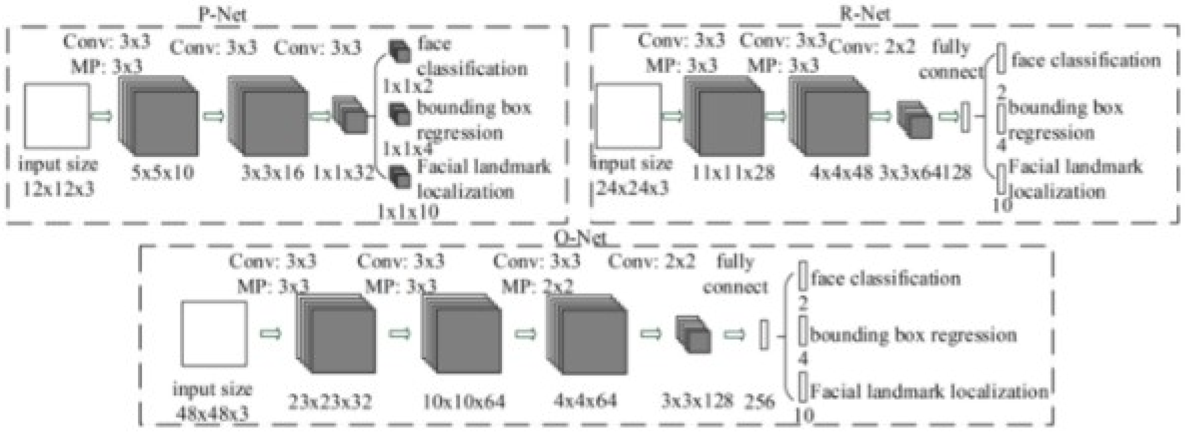
3.1 Face recognition project

While face recognition has been around in one form or another since the 1960s, recent technological developments have led to a wide proliferation of this technology. This technology is no longer seen as something out of science fiction movies like Minority Report. With the release of newly invented technologies, millions of people now literally have face recognition technology in the palms of their hands, protecting their data and personal information. While mobile phone access control might be the most noticeable way face recognition is being used, it is being employed for a wide range of use cases including preventing crime, protecting events and making air travel more convenient. Therefore, we choose this topic for the project.

3.1.1 Problem solving steps

There are 4 steps in processing:

- Loading images from dataset:
  Input data from folders by name of people. Each folder contains personal images by name; and the folder names are automatically used as classes for classification model.
- Detecting faces in images:
  The model used to detect face from an image is MTCNN. That stands for Multi-task Cascaded Convolutional Networks. It composes of 3 stacked CNNs and works at the same time when detecting faces. Each network has a different structure and plays a different role in the task. The output of MTCNN is the position of the face and points on the face such as eyes, nose, mouth.



Structure of MTCNN model

- Pre-processing to embed faces:
  FaceNet being a deep neural network is used to extract features from an image of a personal face. It was published in 2015 by Google researchers Schroff et al.

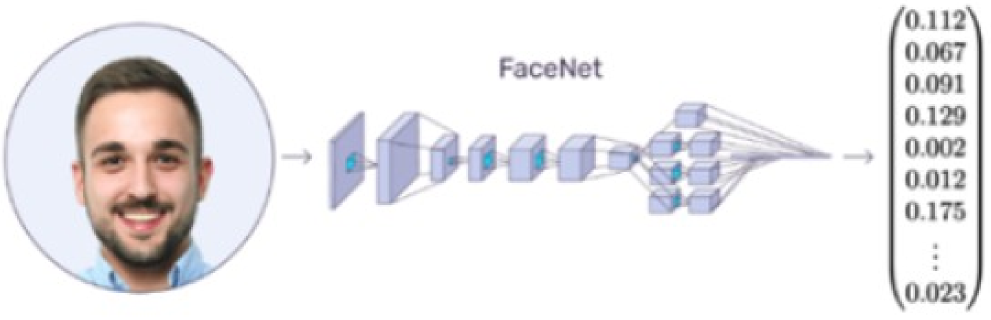


Figure 1: FaceNet embedding

FaceNet takes an image of the person's face as input and outputs a vector of 128 numbers which represent the most important features of a face. In machine learning, this vector is called embedding. Why embedding? Because all the important information from an image is embedded into this vector. Basically, FaceNet takes a person's face and compresses it into a vector of 128 numbers. Ideally, embeddings of similar faces are also similar. Embeddings are vectors and we can interpret vectors as points in the Cartesian coordinate system. That means we can plot an image of a face in the coordinate system using its embedding.

One possible way of recognizing a person on an unseen image would be to calculate its embedding, calculate distances to images of known people and if the face embedding is close enough to embed of person A, we say that this image contains the face of person A.

In addition, for more convenience, Label Encoder is used to encode the labels of the images to numeric value. That helps model works more easily After labels are encoded, they will look like:

Validation labels:

[2 1 0 0 1 0 1 1 0 3 1 0 2 0 3 3 2 0 3 0 3 2 0 1 0 2 3 0 1 1 0 1 0 3 31 1 0 3 1 2 1 0 3 2 2 1 1 1 0 0 2 2 1 0 0 1 3 1 0 1 1 2 1 1 1 0 2 0 3 2 2]

- Training model to recognize whose the faces belong to:

- Support Vector Classification

Definition: Support Vector Machines (SVM) is considered to be a classification approach, but it can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the data set into classes.

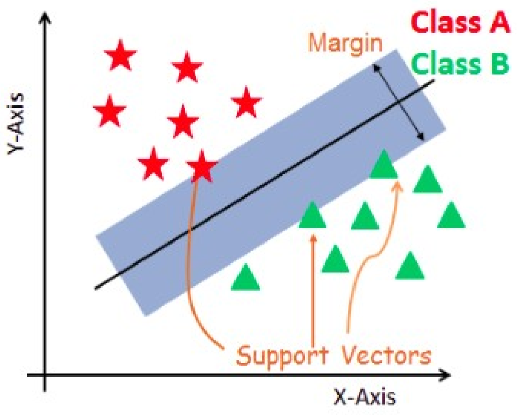


Figure 2: Support Vector Machine

- Advantages:

+ SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handlenonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages classification of genes, and handwriting recognition.

+ SVM classifiers offer good accuracy and perform faster prediction compared to Naive Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

- Disadvantages:

SVM is not suitable for large data sets because of its high training time and it also takes more time in training compared to Naive Bayes. It works poorly with overlapping classes and is also sensitive to the type of used kernel.

- How to use SVC in the program:

model_SVC = SVC(kernel = 'linear', probability = True, tol =1e-3, cache_Size = 200, max_iter = -1)

+ Kernel: linear

+ Tolerance (tolerance for stopping criterion): 10^{-3}

+ Cache size (specify the size of the kernel cache): 200MB

+ Max iteration: -1 (means no limitation)

Logistic Regression

Logistic regression is a statistical method for predicting binary classes. The out come or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

- Advantages:

+ Because of its efficient and straightforward nature, it does not require high computation power, easy to implement, easily interpretable, widely used by data

analyst and scientist. Also, it does not require scaling of features. Logistic regression provides a probability score for observation.
+ Easy to implement, and extend to multiple classes and a natural probabilistic view of class prediction.
+ Fast at classifying unknown records.
- Disadvantages:
Logistic regression is not able to handle a large number of categorical features/-variables. It is vulnerable to over-fitting, also cannot solve the non-linear problem with the logistic regression. That is why it requires a transformation of non-linear features. Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.
- How to use Linear Regression in the program:
model_Linear = linear_model.LogisticRegression(penalty = '12', tol = 1e-4, solver = 'liblinear')
+ Penalty: l2
+ Tolerance (tolerance for stopping criterion): 10^{-3}
+ Solver: liblinear
(For small datasets, 'liblinear' is a good choice, 'sag' are faster for large ones)
Random Forest Classifier
Let us understand the algorithm in layman's terms. Suppose that you want to goon a trip and you would like to travel to a place which you will enjoy. So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends. Let us suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.
In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.
The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forest algorithm.
It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split data set. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. Ina classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered

as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

- Advantages:

+ Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.

+ It does not suffer from the over-fitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

+ The algorithm can be used in both classification and regression problems.

+ Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing value.

+ You can get the relative features importance, which helps in selecting the most contributing features for the classifier.

- Disadvantages:

+ Random forest classifier is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.

+ The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

- How to use Random Forest Classification in the program:

model_RandomForest = ensemble.RandomForestClassifier(n_estimators = 5, min_impurity_decrease = 0.05, max_depth = 13)

+ Number of estimator: 5 (Higher number of estimator does not give better result)

+ Minimum of impurity decrease: 0.05

+ Max depth: 13 (There are only 17 people in this problem, there should not be large number of features. Therefore, each tree does not over-fit the model the give the same vote while training)

+ Bootstrap: true (as default)

3.1.2 Evaluation and Conclution

Support Vector Classification

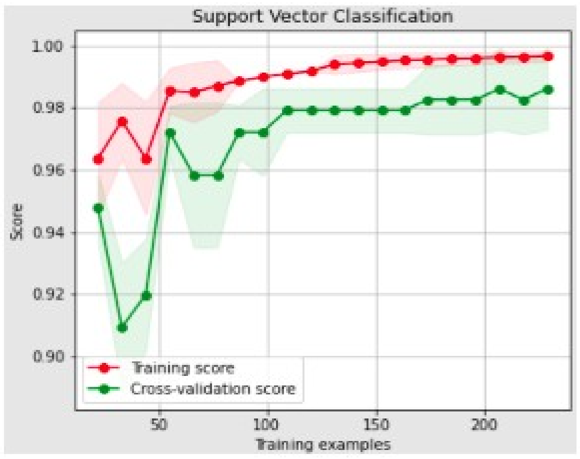


Figure 3: Training progress of Support Vector Classification model

In Support Vector Classification model, the model tries to learn image by image, which increase the score little by little. The initial score is not so high because the model's initialized parameters are not so good. This is reasonable because its initialized parameters are normally randomly generated. After some iterations, the model starts learning better and the score increase gradually.
The learning curve is illustrated at figure 3.
My evaluation achieves an accuracy of 100%.

Logistic Regression

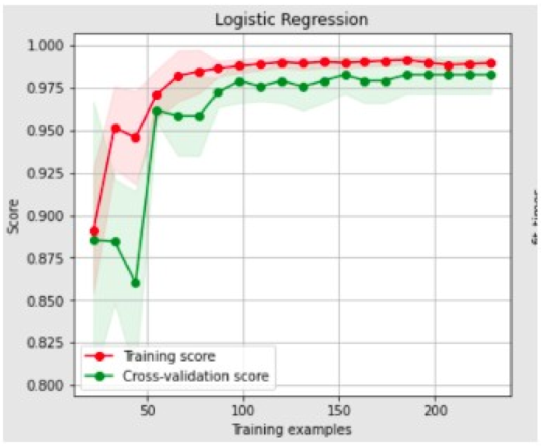


Figure 4: Training progress of Logistic Regression model

In Logistic Regression model, the model learns from the dataset gradually and it quickly achieves high accuracy score because of the simplicity of my dataset. The cross-validation score is a bit lower than the training score, which means that my model learns pretty well and does not over-fit the training dataset.
The learning curve is illustrated on the figure 4.
My evaluation achieves an accuracy of 97.2%.

Random Forest Classifier

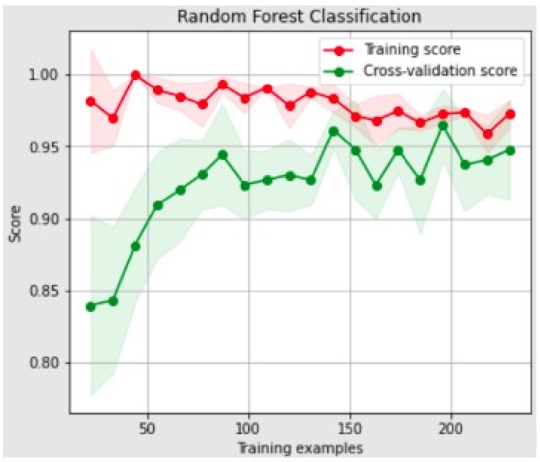


Figure 5: Training progress of Random Forest Classifier model

In Random Forest Classifier, the model quickly over-fits the training dataset because it is so powerful when the training dataset is small. At this stage, the model achieve pretty low accuracy score.
After that, when the number of tree (specified by n_estimators reaches, the forest cannot be extended anymore. At this stage, the model will try to fit new coming images with current forest. This makes the model converges at a state that it no longer over-fits the training dataset. Hence, the cross-validation score increases significantly even though the training score decreases slightly.
The learning curve is illustrated on the figure 5.
My evaluation achieves an accuracy of 93.1%.

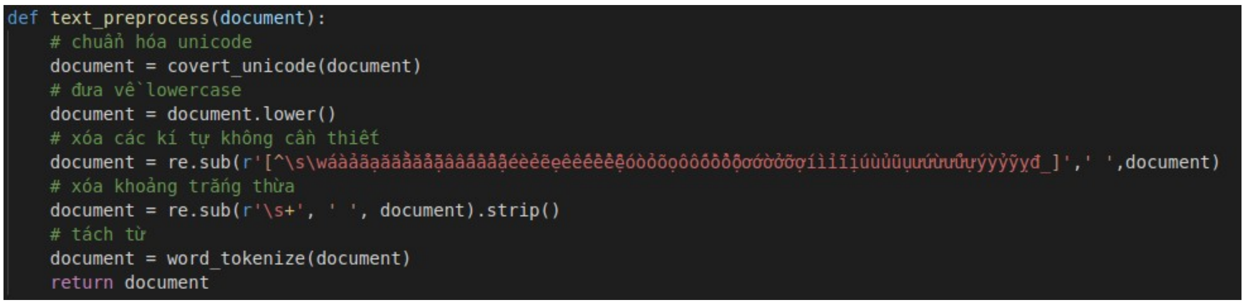
Conclusion
From the experimental results above, It can be seen that all three models learn pretty well on my dataset. Even though there are some little differences in validation accuracy of them, these distinctions are insignificant. In order to obtain a result with clearer differences between more, I need to build a dataset with more images with variety of image-capturing condition. Furthermore, I need to adjust and choose some more appropriate parameters so that my models converge as expected in a time-and-memory-limited condition.

3.2 Text Classification project

Text classification is a machine learning technique that assigns a set of predefined categories to open-ended text. Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web.
For example, new articles can be organized by topics; support tickets can be organized by urgency; chat conversations can be organized by language; brand mentions can be organized by sentiment; and so on.

3.2.1 Problem solving steps

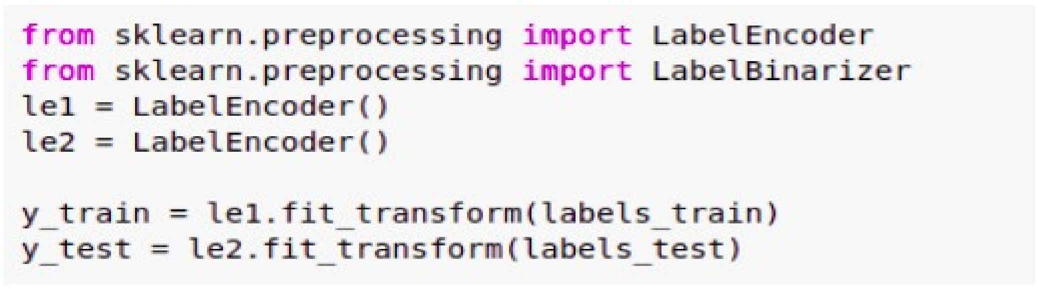
There are 4 steps in processing:
- Data preprocessing:



```
def text_preprocess(document):
    # chuẩn hóa unicode
    document = covert_unicode(document)
    # đưa về lowercase
    document = document.lower()
    # xóa các kí tự không cần thiết
    document = re.sub(r'[^\s\wáàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩịúùủũụưứừửữựýỳỷỹỵđ_]',' ',document)
    # xóa khoảng trắng thừa
    document = re.sub(r'\s+', ' ', document).strip()
    # tách từ
    document = word_tokenize(document)
    return document
```

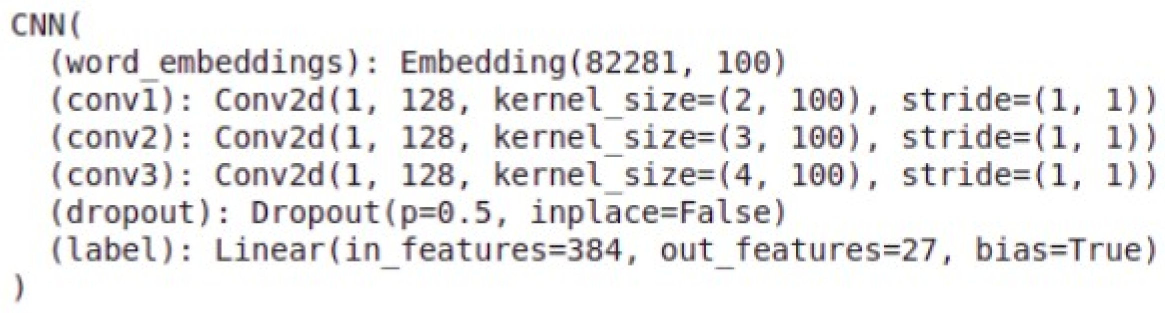
Figure 6: Data preprocessing
- Word embedding:

- Word2Vec and FastText representation methods are used to represent words as list of numbers (100 numbers)
- Using a dictionary named w2i (word to index) to install keys that are words in train text dataset; and values are the corresponding ordinal number of the keys in the dict. There are 2 special keys:
+ "<pad>": represents the "pad" positions in the text after padding.
+ "<unk>": represents words that are not included in the training set texts.
- Using a dictionary named i2w (index to word): store the opposite value of w2i.
- Using a list named vocab_embed with a number of rows equal to len(w2i) and a number of columns of 100 (the length of a list representing a word); meaningful at the i-th position is the numeric list representing the i-th word in i2w.
- Using vocab_embed list to create id_train and id_test that are two lists containing numeric lists representing documents, including lists (length 100) representing words in train set and words in test set.
=> Each document is represented as a list of n (n is the number of words extracted from the text) list of length 100.
- Encoding labels of documents to number type:



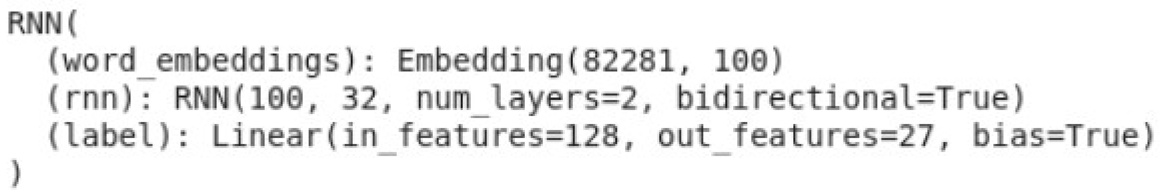
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelBinarizer
le1 = LabelEncoder()
le2 = LabelEncoder()

y_train = le1.fit_transform(labels_train)
y_test = le2.fit_transform(labels_test)
```

- Packing data as batches:
  Setting batch_size = 32 means that there are 32 input texts and corresponding labels in each batch. Since the data set is too large, we cannot put all the data set in for training at a time. Force us to split the data set into batches (smaller size).
- Training:
  CNN model structure



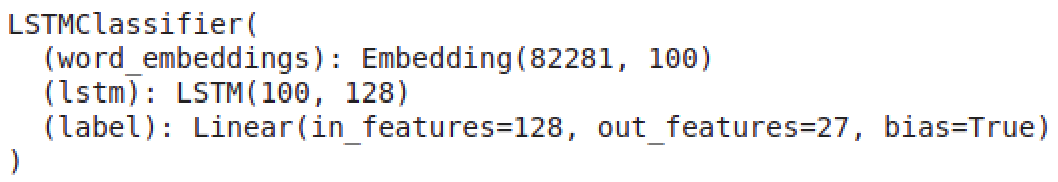
```
CNN(
  (word_embeddings): Embedding(82281, 100)
  (conv1): Conv2d(1, 128, kernel_size=(2, 100), stride=(1, 1))
  (conv2): Conv2d(1, 128, kernel_size=(3, 100), stride=(1, 1))
  (conv3): Conv2d(1, 128, kernel_size=(4, 100), stride=(1, 1))
  (dropout): Dropout(p=0.5, inplace=False)
  (label): Linear(in_features=384, out_features=27, bias=True)
)
```

82281 is the value of the length of the word vocabulary.
RNN model structure



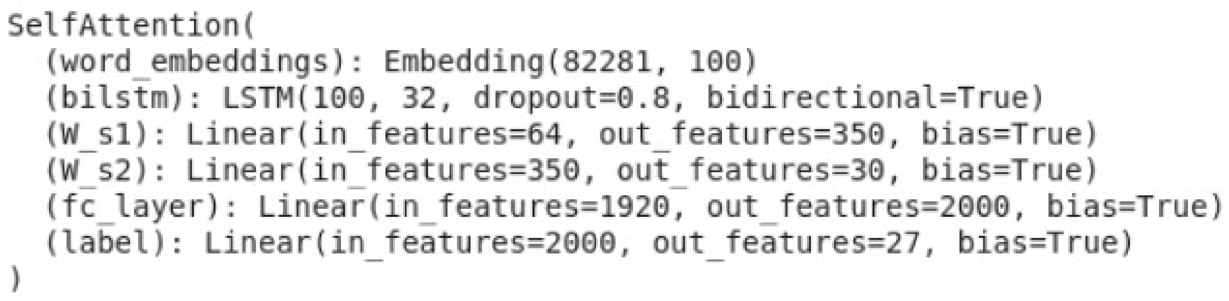
```
RNN(
  (word_embeddings): Embedding(82281, 100)
  (rnn): RNN(100, 32, num_layers=2, bidirectional=True)
  (label): Linear(in_features=128, out_features=27, bias=True)
)
```

LSTM model structure



```
LSTMClassifier(
  (word_embeddings): Embedding(82281, 100)
  (lstm): LSTM(100, 128)
  (label): Linear(in_features=128, out_features=27, bias=True)
)
```

Selfattention model structure



```
SelfAttention(
  (word_embeddings): Embedding(82281, 100)
  (bilstm): LSTM(100, 32, dropout=0.8, bidirectional=True)
  (W_s1): Linear(in_features=64, out_features=350, bias=True)
  (W_s2): Linear(in_features=350, out_features=30, bias=True)
  (fc_layer): Linear(in_features=1920, out_features=2000, bias=True)
  (label): Linear(in_features=2000, out_features=27, bias=True)
)
```

3.2.2 Evaluation and Conclusion

Evaluation

Model	Word2Vec	FastText
CNN	81.48%	81.12%
RNN	65.49%	64.62%
LSTM	71.39%	81.99%
SelfAttention	76.43%	85.08%

Conclusion

In general, the representation model from FastText gives better accuracy results than the Word2Vec model; that is clearly shown when training with LSTM and Self Attention models. The reasons are:

- Word2Vec cannot detect words that are not in the training dataset. Word2Vec uses different vectors to represent words in different forms.
- The vectors obtained from FastText are better for many less common compound words, even for words not present in this dictionary by referencing other words with similar structure.