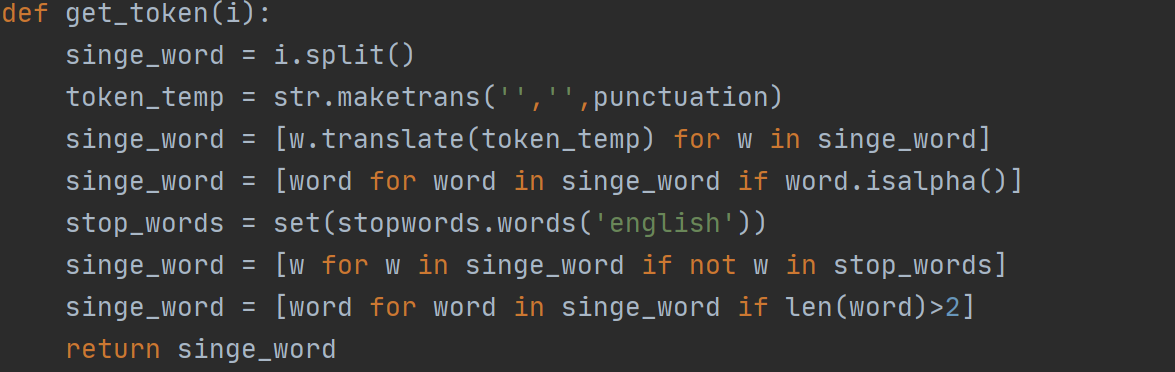
**CNN with Word2Vec embeddings Model**

**Preprocessing**

1. Read all the file and restore to .csv file, then read the .csv file again.
2. Divide each text into words and remove all the punctuations.
3. Remove stop words which cannot reflect the characteristics of the text
4. Set up random seed, randomly divided test set and training set ()



**Word2Vec Model**

Filter out the words that meet the above requirements in each training set text and convert these keywords into feature vectors. In order to select the features of the text, we can treat each word of the text as a feature, and we can think of using one-hot encoding to represent the feature. However, there's a problem with that the number of key words in a text is hundreds or thousands, so the dimension and calculation amount of the generated matrix will be very large. As a result, the parameters that need to be adjusted in the hidden layer of the neural network training model will be greatly increased in the later stage.

**Language Work Mechanism**

Additionally, it cannot express the relation of words to words. So it's just a very inefficient method. Word2Vec model can support to solve this problem which can show the relationship between words and integrate word vectors by similarity to reduce the dimensionality further. The Feedforward Neural Net Language Model (NNLM) is used to predict context, the word that predicts the position based on the first words.

: shape of one-hot vector of vector and get the shape vector.

: A fully connected layer with as activation function.

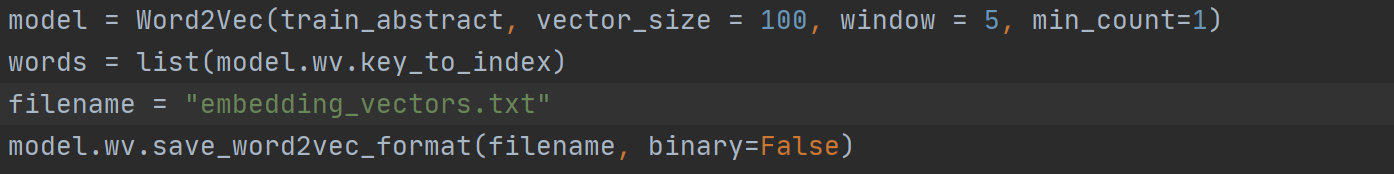
: A fully connected layer which followed by a SoftMax function produces a probability distribution.

In order to improve the precision, this model introduce the Language Confusion:

Combine with the cross-entropy loss function, each time we can update the better result and implement unsupervised learning. Assume is Number of sentence words :

To conclude, through NNLM we can solve Synonym problem (polysemy). In the output layer, SoftMax function can support speeding up the training.

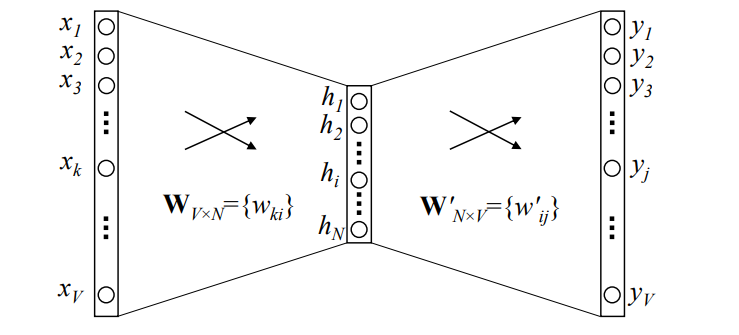
**Principle of Word2Vec**



The core thinking of the Word2Vec is using words to predict words. There are two main models and we just select the Skip-Gram model which is a little different from Language Model, using to predict context by the head words.

The probability of predicting the next word can be expressed as:

According by the *word2vec parameter learning explained*, the activation function of the hidden layer is actually linear, which is equivalent to no process. The model is trained by the neural network to use the back propagation algorithm. In the weight of the hidden layer, only the weight corresponding to value 1 is activated. As a result, we can get the weight of the neural network as output, so we can get the relationship between words. At the same time we can obtain the lower dimension.



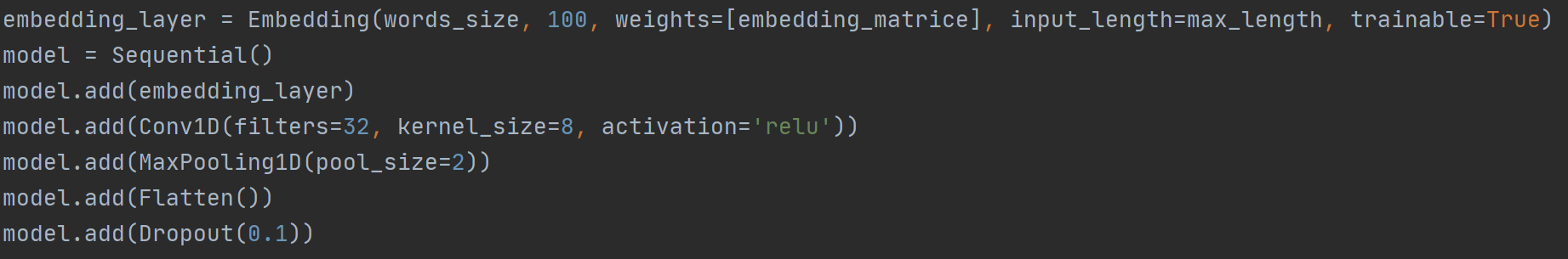
Then, Save the trained model to local. The first line is dimension and remainder are word vectors.



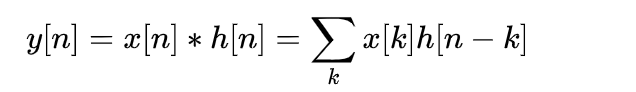
**Classifier model**

1. : Building a vocabulary
2. tokenizer: Transform word lists into word index sequences (dictionaries)
3. : In order to facilitate the implementation, the framework can only accept input of sequences of the same length, so the sequence needs to be filled to make all texts in the text set have the same length.
4. to categorical: Converts the categorical label to one-Hot encoding
5. Load the embedding and transform it to a matrix. Matrix operations will be much faster than other forms of calculations.

**CNN Model**



1. Generating embedding layer as input layer, set the vocabulary size and the output size which must be identical with the Word2Vec parameter. And the weight of each word is expressed by the embedding matrix.
2. Convolution layer: The convolution layer retains the main features while reducing the parameters and computation to improve the model generalization ability. In our task for text classification, text is just a sequence type which can be resolved by the A one-dimensional convolution. The function can be written as:



is the our input (word vector) and the is response signal (convolution kernel). We start with the model's default convolution kernel. Here we set the

* The reason why convolution requires an activation function:

Convolution can be understood as assigning a weight to each item of the filter, and this operation is linear. All are not necessarily linearly separable for our samples, so the expressive power of the linear model is not enough, so the activation function must be introduced.

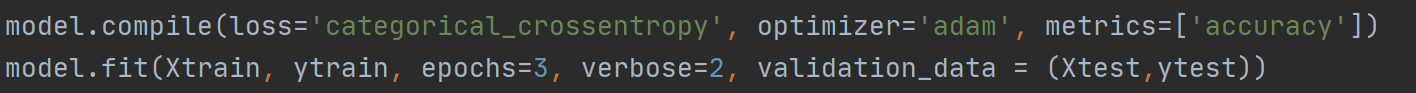
* The reason why the activation function is :

We do not use , and other activation function. Because there's a lot of multiplication and division involved in doing the activation function with a large amount of computation. While activation function is used to save a lot of calculation in the whole process. It tries to represent text features with as many zeros as possible. As a result, it generates a sparse matrix which can remove the redundancy in the data and retain the features as much as possible. In addition, in order to simulate more detailed changes, the output value can be more than 0 to 1. So we do not need to normalize. Thus, the whole process saves a lot of calculation.

1. Pooling layer: We use the maximum type pooling layer which can improve fault tolerance of the model and further feature sampling and dimensionality reduction, reduce model size as well as improve calculation speed. In addition, it can reduce the over-fitting probability and improve the robustness of feature extraction.
2. Output layer: Use fully connection with 4 layers categories and the activation function is SoftMax function.
3. Loss function: Exchange entropy loss function and we set the number of convergences to 3.

**Result Test**

Fit the model by the test set, the result accuracy can reach more than .



**↓**

