Patent Classification with Deep Learning

Skim-gram, Deep Learning

1 Summary

The aim of this report is to show how to perform patent classification on the patent USPTO Data Set via two models, one for learning embeddings and representation (skim gram model) and the other to perform classification (a convolutional neural net) (based on the DeepPatent: patent classification with convolutional neural networks and word embedding research paper).

The code performs preprocessing removing foreign languages and reconstructing the data in a document formatted appropriately to enter the models.

A short discussion is delivered in regards to model performance and evaluation as well as the respective cost function and noting also that the model achieved 61% accuracy, however the validity of this is questioned since the data set given is small and not giving enough room for training or generalization.In addition the number of epochs used for training where small and hence the model was not trained sufficiently, but that is only due to computational reasons

Finally the case of continual learning is considered with catastrophic forgetting for the case when new data are constantly arriving as well as a threshold for retraining.

2 Question 1

2.1 Discuss the performance of your model.

In order to perform model two fundamentals elements are required. First the data set has to be split in three categories, that is train,validation and test. The first two are obtained from overall data set, forming train-validation set and a test set (usually a 20% split and 80% split on validation and train). Afterwards in the train-validation set a K-cross validation optimization takes place along with repeated experiments at each fold. To elaborate, consider the case where K=5, then a 5-cross validation takes place as shown in the figure below:

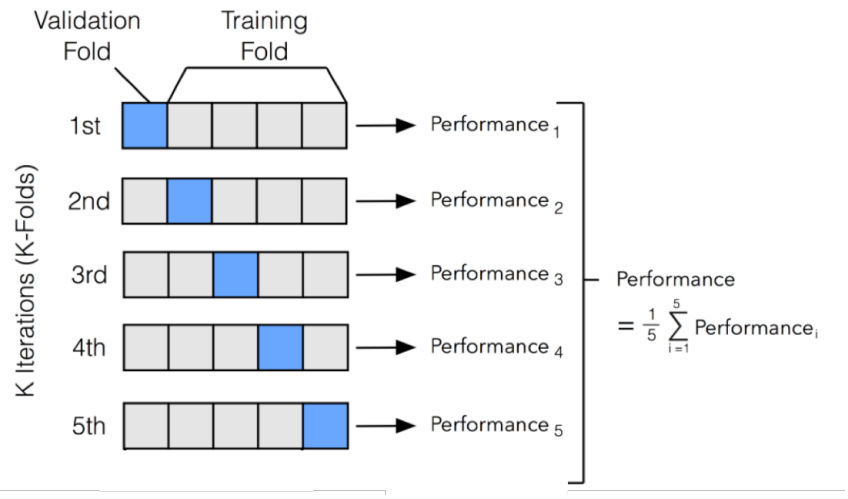

Figure 1: An illustration of K=5 - cross validation, indicating the split between validation and train data

For the case of K=5 there are 5 different possible sets that the training data can come from, which in turn it trains the model and then it is evaluated on remaining data (the validation set). This is usually repeated a couple of times in order to obtain the statistics of the model performance.

In general in addition to that there is a hyperparameter search that is coupled with some method, such as random sampling, Bayesian optimization and grid search. The simplest being grid search in which case the K cross validation methodology is repeated for each hyperparameter setting (as defined by the user) and then based on average performance on the validation set the best hyperparameter values are chosen. In the case of a neural network possible hyperparameters can be a regularization constant (this is used to avoid over-fitting) or number of neurons in the architecture that can also control the model complexity.

In this case however the data set is small (358), but since 300,000 thousand patents are coming each year the model must be able to scale and perform well in large number of data points. More specifically, firstly pre-processing the data set takes place, that is the all the patent information are considered to be part of a single document and hence the id, title, filling date, patent-authors and summary are concatenated. Afterwards each document is 'normalized' in the sense that stop words are removed and then a respective vocabulary is formed in order to train the skim gram model. The skim gram model was used (note this is unsupervised learning algorithm) in order to extract the context of the patent-document and then afterwards it is fed into a convolution neural network in order to to classify the document.

In this case however, due to the time needed to train both models only a train-test split was performed to evaluate the model. In addition in order to avoid over-fitting early stopping is used at ten data points apart, however this hyperparameters is not searched due to the need for training of each model. The idea of early stopping is illustrated below:

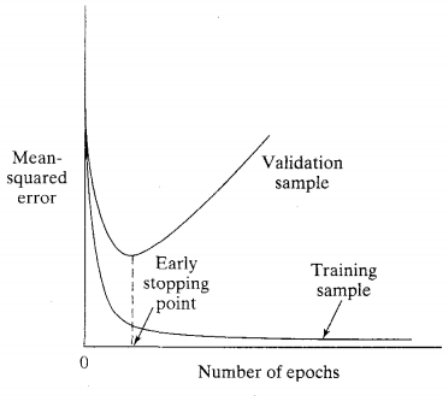

Figure 2: The figure captures the early stopping rule

2.2 Discuss the methodology you used to evaluate your model.

In this case the skim gram model is evaluated on the test set by seeing how close the out of sample words are correctly captured by the skim gram model. However in general more details are required since the outputs are probabilities and hence it is not appropriate to evaluate in this manner,more specifically a Q&A data set needs to formed in order to properly evaluate the accuracy of the skim gram model, for example 'a is to b as c is to _?'. Finally it should be noted that the data set is short and hence pre-trained model could have better performance however consider the goal of this exercise it was chosen to restrict the models only to the given data set. (please see the appendix to see more details on the skim gram model).

The model that applies document classification is evaluated using cross entropy, but for one class instead. More specifically considering that this is a classification problem it is logical to view the error function as the maximum likelihood of Bernoulli distribution since there can be two classes (Patent or no Patent). In other words the error has to minimize the negative likelihood (this is the cross entropy error function):

$$E = -\ln(\mathcal{L}) = -\sum_{p}(1 - targ_p * \ln(1 - out_p)) + (targ_p * \ln(out_p)) \tag{2.1}$$

Where $targ_p$ is the target output and out_p is the network output for the data point with index p. In addition in order to avoid overfitting the early stopping mechanism was used , see figure 2 . To elaborate, after each epoch the error is evaluated with the validation set. If the validation error is smaller than a previous validation error value then the corresponding weight values are recorded. Afterwards training proceeds for some user defined amount of epochs (called waiting), if the validation error has not reduced at all within the latter epochs then training terminates. Note that the algorithm continues to train for some epochs in order to make sure that the validation error attained is not because of noise or local instability.

The evaluation of the overall performance happened over the test data that where split from the overall training data. The classification is accurate the if softmax output value is closer to the true label value (either close 0 or 1)
The results are summarized in the plots below, please note the test accuracy is not high but that is not a good way to validate the model as the sample size is too small.In addition the number of epoch to train the model and skim gram was small due to computational reasons.

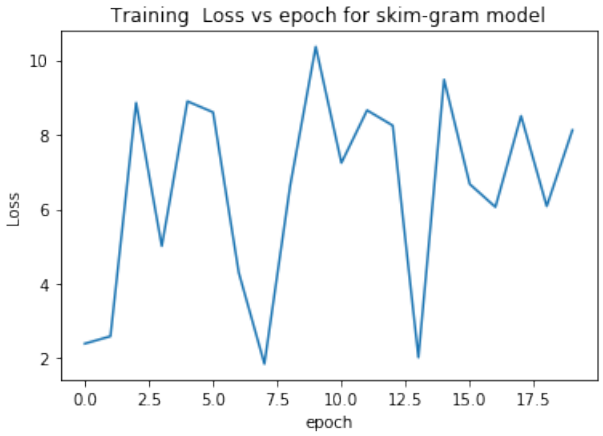

Figure 3: The figure shows the skim gram model loss function for training data

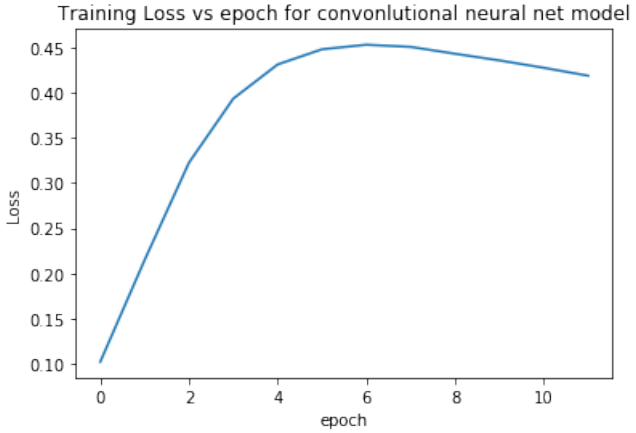

Figure 4: The figure shows the train loss for the convolutional neural net (the classifier)

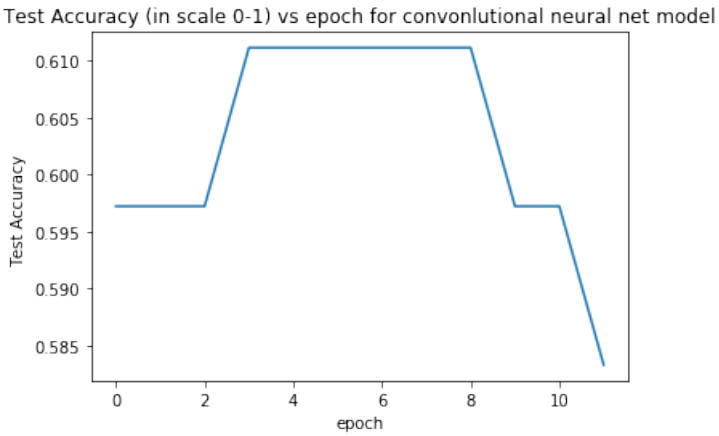

Figure 5: The figure shows the test accuracy for the convolutional neural net (the classifier)

3 Question 2

3.1 How will the system decide when to update the model?

The biggest issue with capturing new data is how will the model accuracy will behave over time? Will there be any drift in the accuracy of the model as new data are obtained?. The issue to be dealt with, requires to monitor the model as new data are arriving. It is important first to define a threshold value on the model performance such that when the performance of the model reduces below that point, the model will be retrained.

To elaborate, the simplest check is to set a threshold value (this depends on the application) and when the accuracy of the model is lower, then retrain the model from the beginning including the newly obtained data. In addition, to that one can train the model online as new data points are obtained with hope to reduce the frequency of retraining. However both issues do not solve the problem of catastrophic forgetting which can be important when new data are constantly coming. Consequently even after retraining it is not possible for the model to capture the well the entire population.

In general there are several methods that exist to deal with this in the literature, but one simple approach comes from continuous learning and a simple way to apply this via statistical leveraging with SVD in order to select the order of the data that retrains the model (the details for that can be found in Continual Learning via Online Leverage Score Sampling).The algorithm for that is described as shown in the paper below:

Algorithm 1 Online Leverage Score Sampling

Input: A sequence of tasks $\{(A_1, B_1), ..., (A_i, B_i), ...\}$ with $A_i \in \mathbb{R}^{n_i \times d}$ and $B_i \in \mathbb{R}^{n_i \times m}$; initialization of the model parameters; a space parameter ℓ i.e., number of samples to pass in the model for training. It can be set as n_i or even smaller after receiving the i-th task, which avoids extra memory and computations during training.

Output: A trained neural network on a sequence of tasks.

Initialize a buffer set $S = \{\hat{A}, \hat{B}\}$ where both $\hat{A}$ and $\hat{B}$ are empty.
while the ith task is presented **do**
    if $\hat{A}$ and $\hat{B}$ are empty **then**
        set $\hat{A} = A_i$ and $\hat{B} = B_i$,
    else
        set $\hat{A} = \begin{bmatrix} \hat{A} \\ A_i \end{bmatrix}$ and $\hat{B} = \begin{bmatrix} \hat{B} \\ B_i \end{bmatrix}$.
    Perform SVD: $[U, \Sigma, V^T] = svd(\hat{A})$.
    Randomly select ℓ rows of $\hat{A}$ and $\hat{B}$ without replacement based on probability $\|U_{j,:}\|_2^2 / \|U\|_F^2$ for $j \in \{1, ..., n_i + \ell\}$ (or $j \in \{1, ..., n_i\}$ when $i = 1$) and set them as $\hat{A}$ and $\hat{B}$ respectively.
    Train the model with $\hat{A} \in \mathbb{R}^{\ell \times d}$ and $\hat{B} \in \mathbb{R}^{\ell \times m}$.

Figure 6: The pseudocode for continuous learning and avoiding catastrophic forgetting

4 Schematics

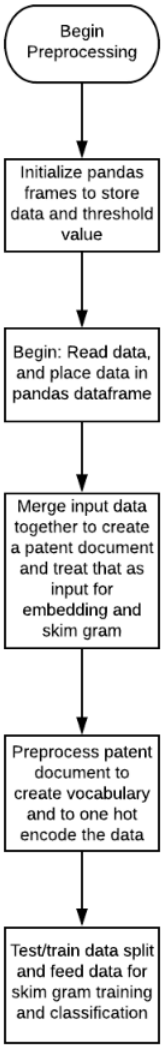

Figure 7: The generic structure of the code

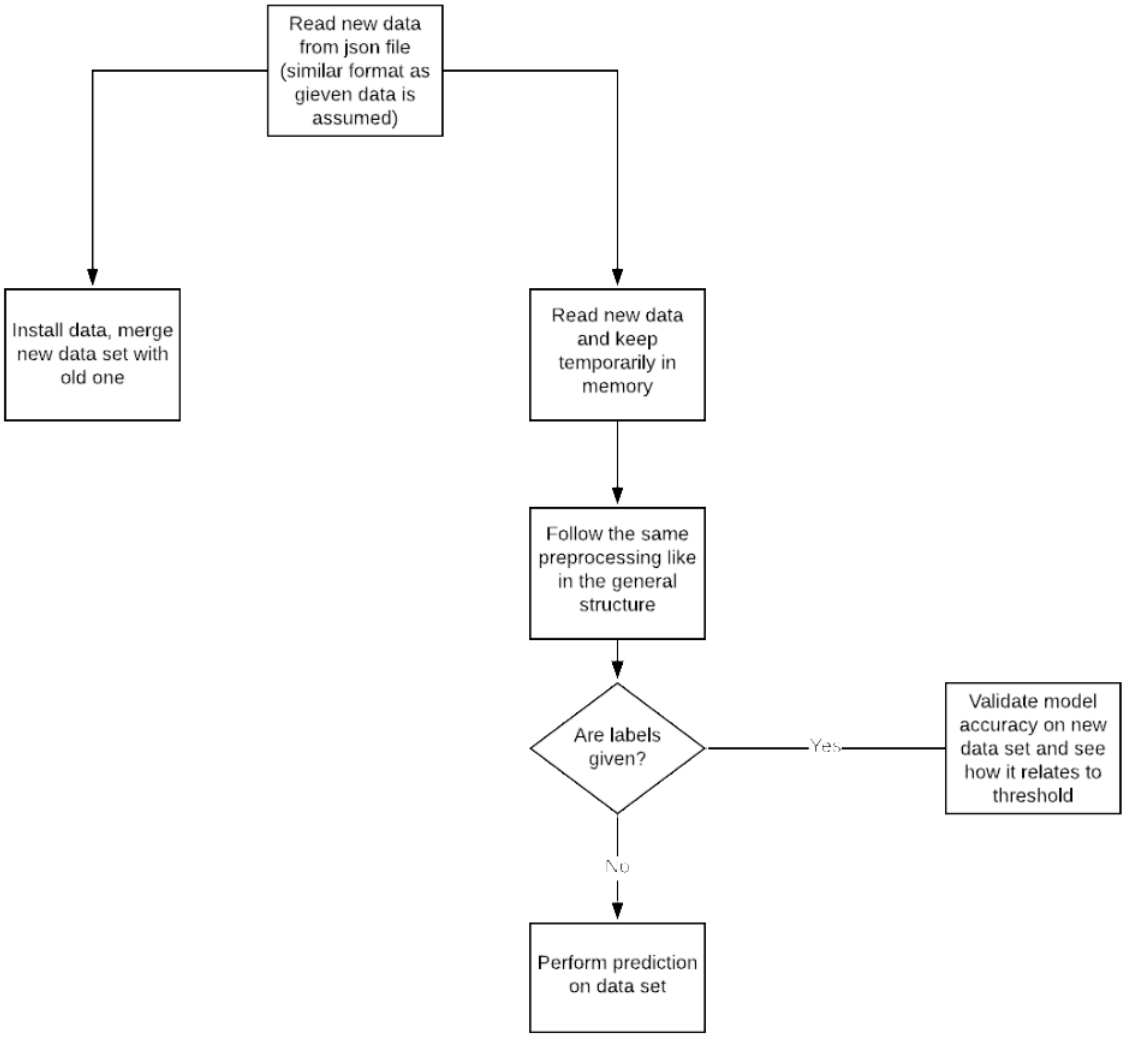

Figure 8: The generic structure of the code for new incoming data

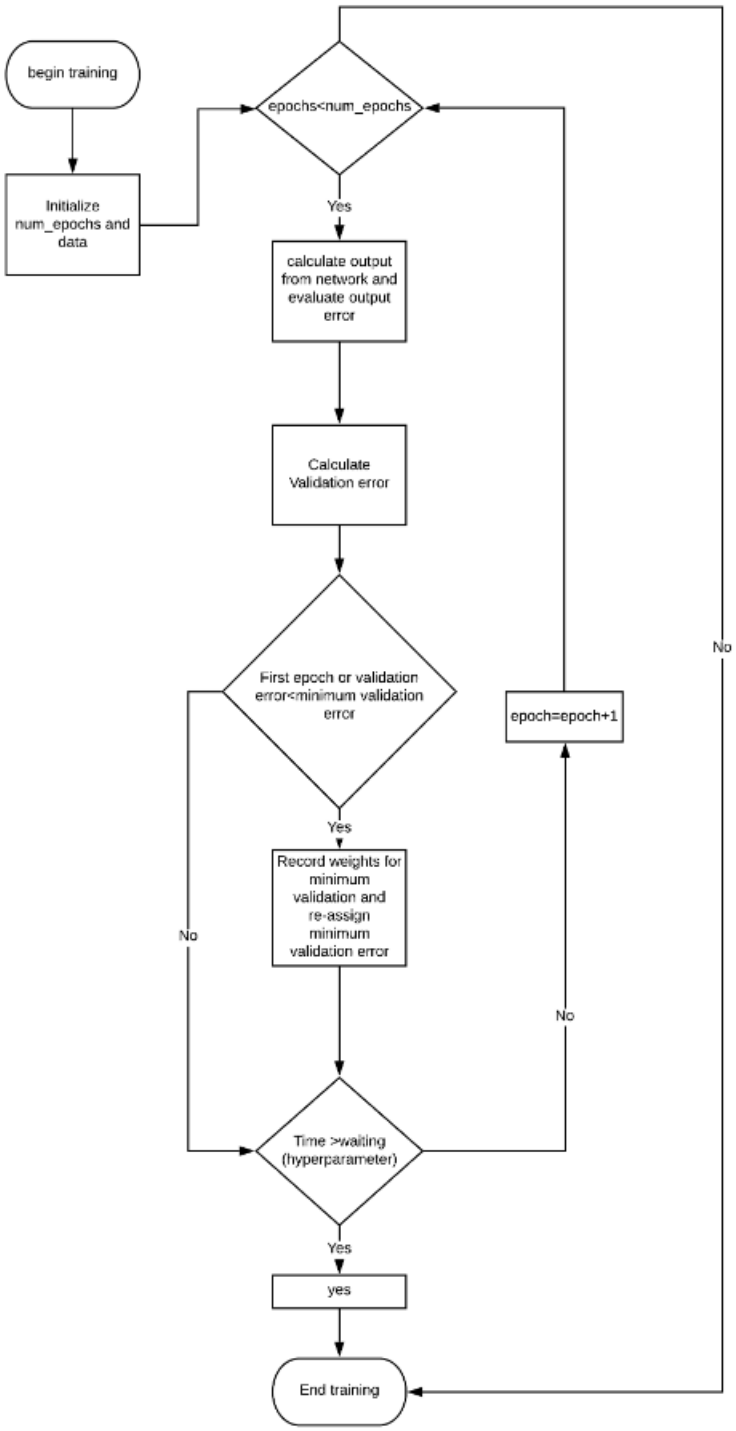

Figure 9: The generic structure of the code for training with early stopping

5 Appendix

5.1 Skim gram

Notation for Skip-Gram Model:

- w_i: Word i from vocabulary V
- $\mathcal{V} \in \mathbb{R}^{n \times |V|}$: Input word matrix
- v_i: i-th column of $\mathcal{V}$, the input vector representation of word w_i
- $\mathcal{U} \in \mathbb{R}^{n \times |V|}$: Output word matrix
- u_i: i-th row of $\mathcal{U}$, the output vector representation of word w_i

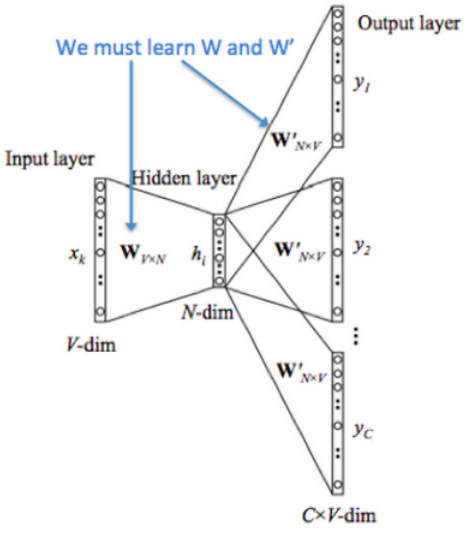

Figure 10: Skim gram architecture

There are several steps to follow for the skim-gram model. Firstly formulate the vocabulary that one hot encodes each word in the data, which in turn is represented as the input. The first layer of the neural network form the embedding of the center word via matrix multiplication. The second layer generates the score values of the neighbouring words based on their location in the window. Afterwards the scores are normalized via softmax. Finally a cross entropy loss is used as cost function the equation below puts the words into math:

$$E = -log(P(w_{c-m}...w_{c-1}, w_{c-1}...w_{c+m}|w_c)) = -log \prod_{j=0, j\neq m}^{2m} P(w_{c-m+j}|w_c) = - \sum_{j=0, j\neq m}^{2m} logP(w_{c-m+j}|w_c) \tag{5.2}$$

Where c is the center word to extract context, w is the window size,w is the word and since this is a multi-classification problem the probabilities give rise to cross-entropy loss for multiple classes in this case

5.2 Classification Net

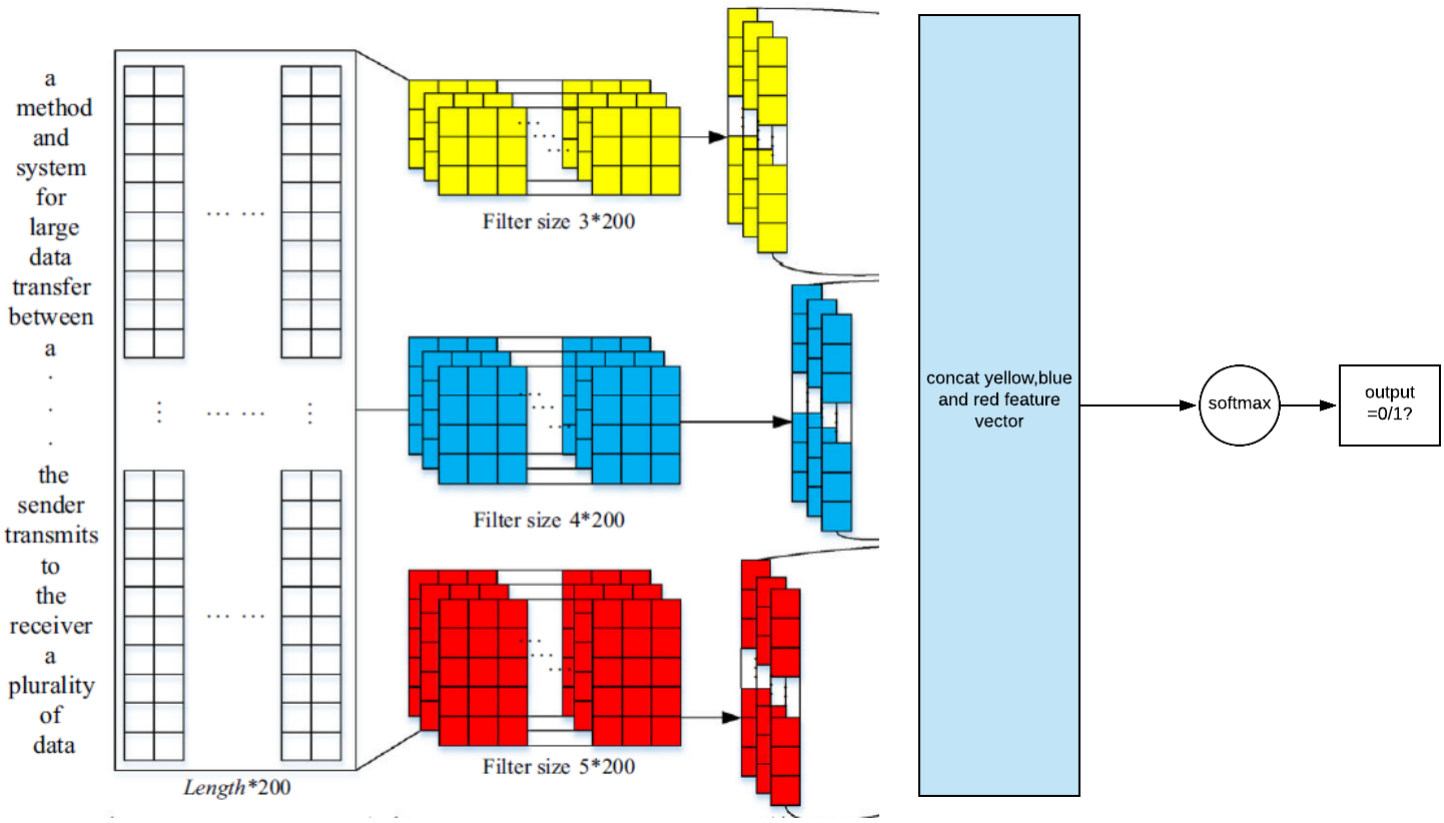

Figure 11: Classification net based on, DeepPatent: patent classification with convolutional neural networks and word embedding research paper