

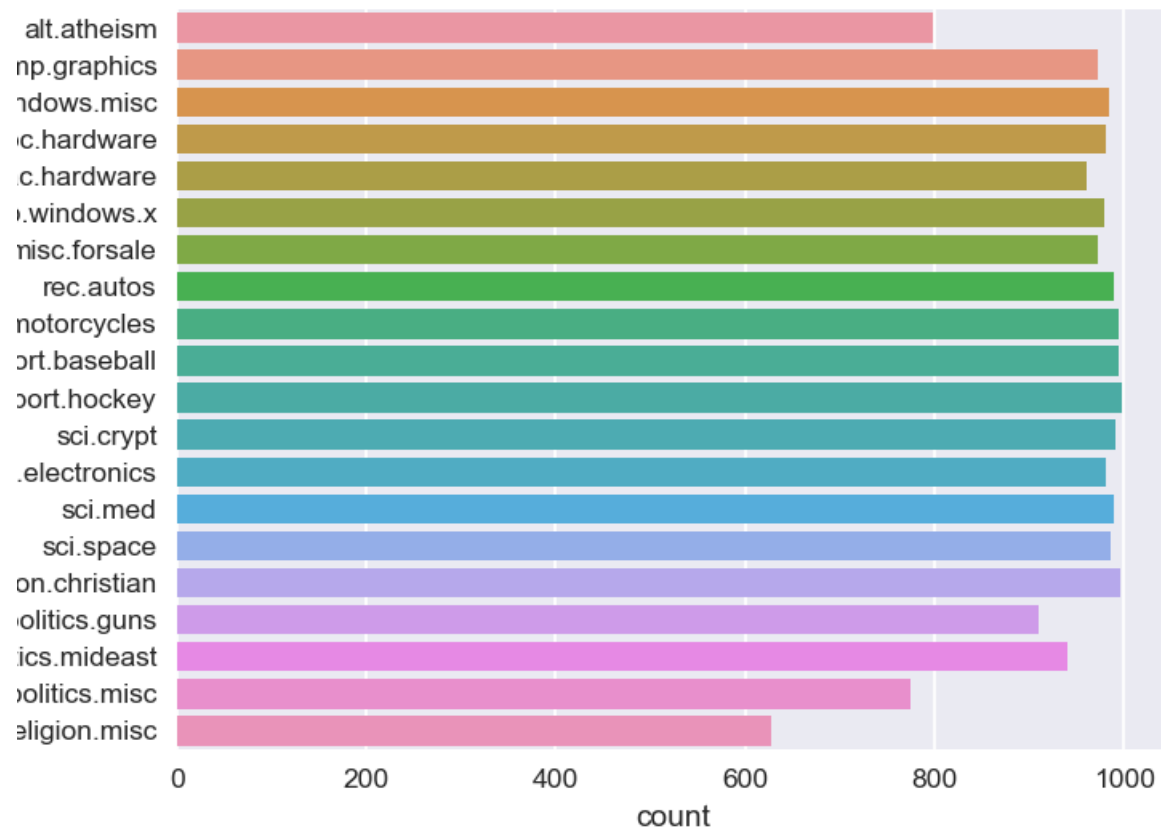
Text Classification:

Data ¶

1. we have total of 20 types of documents(Text files) and total 18828 documents(text files).
 2. You can download data from this [link \(https://drive.google.com/open?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM\)](https://drive.google.com/open?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM), in that you will get documents.rar folder.
If you unzip that, you will get total of 18828 documents. document name is defined as 'ClassLabel_DocumentNumberInThatLabel'.
- so from document name, you can extract the label for that document.
4. Now our problem is to classify all the documents into any one of the class.
 5. Below we provided count plot of all the labels in our data.

```
In [0]: 1 ### count plot of all the class labels.
```

<IPython.core.display.Javascript object>



Assignment:

sample document

Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>

>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now. And there is no "alternative", but the point
>is, "rationality" isn't an alternative either. The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them.

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--

Have you washed your brain today?

Preprocessing:

useful links: <http://www.pyregex.com/> (<http://www.pyregex.com/>).

1. Find all emails in the document and then get the text after the "@". and then split those texts by '.'

after that remove the words whose length is less than or equal to 2 and also remove 'com' word and then combine those words by space.

In one doc, if we have 2 or more mails, get all.

Eg: [test@dm1.d.com, test2@dm2.dm3.com] --> [dm1.d.com, dm3.dm4.com] --> [dm1,d,com,dm2,dm3,com] --> [dm1,dm2,dm3] --> "dm1 dm2 dm3"

append all those into one list/array. (This will give length of 18828 sentences i.e one list for each of the document).

Some sample output was shown below.

> In the above sample document there are emails [jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mango@c s.umd.edu]

preprocessing:

[jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mango@cs.umd.edu] ==> [nyx cs du edu mimsy umd edu cs um d edu] ==>

[nyx edu mimsy umd edu umd edu]

2. Replace all the emails by space in the original text.

```
In [0]: 1 # we have collected all emails and preprocessed them, this is sample output
        2 preprocessed_email
```

```
Out[28]: array(['juliet caltech edu',
                'coding bchs edu newsgate sps mot austlcm sps mot austlcm sps mot com dna bchs edu',
                'batman bmd trw', ..., 'rbdc wsnc org dscomsa desy zeus desy',
                'rbdc wsnc org morrow stanford edu pangea Stanford EDU',
                'rbdc wsnc org apollo apollo'], dtype=object)
```

```
In [0]: 1 len(preprocessed_email)
```

```
Out[29]: 18828
```

3. Get subject of the text i.e. get the total lines where "Subject:" occur and remove the word which are before the ":" remove the newlines, tabs, punctuations, any special chars.
Eg: if we have sentence like "Subject: Re: Gospel Dating @ \r\r\n" --> You have to get "Gospel Dating"
Save all this data into another list/array.

4. After you store it in the list, Replace those sentences in original text by space.

5. Delete all the sentences where sentence starts with "Write to:" or "From:".

> In the above sample document check the 2nd line, we should remove that

6. Delete all the tags like "< anyword >"

> In the above sample document check the 4nd line, we should remove that "< 65882@mimsy.umd.edu >"

7. Delete all the data which are present in the brackets.

In many text data, we observed that, they maintained the explanation of sentence or translation of sentence to another language in brackets so remove all those.

Eg: "AAIC-The course that gets you HIRED(AAIC - Der Kurs, der Sie anstellt)" --> "AAIC-The course that gets you HIRED"

> In the above sample document check the 4nd line, we should remove that "(Charley Wingate)"

8. Remove all the newlines('\n'), tabs('\t'), "-", "\".

9. Remove all the words which ends with ":".

Eg: "Anyword:"

> In the above sample document check the 4nd line, we should remove that "writes:"

10. Decontractions, replace words like below to full words.
please check the donors choose preprocessing for this

Eg: can't -> can not, 's -> is, i've -> i have, i'm -> i am, you're -> you are, i'll --> i will

There is no order to do point 6 to 10. but you have to get final output correctly

11. Do chunking on the text you have after above preprocessing.

Text chunking, also referred to as shallow parsing, is a task that follows Part-Of-Speech Tagging and that adds more structure to the sentence.

So it combines the some phrases, named entities into single word.

So after that combine all those phrases/named entities by separating "_".

And remove the phrases/named entities if that is a "Person".

You can use `nltk.ne_chunk` to get these.

Below we have given one example. please go through it.

useful links:

<https://www.nltk.org/book/ch07.html> (<https://www.nltk.org/book/ch07.html>).

<https://stackoverflow.com/a/31837224/4084039> (<https://stackoverflow.com/a/31837224/4084039>).

<http://www.nltk.org/howto/tree.html> (<http://www.nltk.org/howto/tree.html>).

<https://stackoverflow.com/a/44294377/4084039> (<https://stackoverflow.com/a/44294377/4084039>).

In [0]:

```
1 #i am living in the New York
2 print("i am living in the New York -->", list(chunks))
3 print(" ")
4 print("-"*50)
5 print(" ")
6 #My name is Srikanth Varma
7 print("My name is Srikanth Varma -->", list(chunks1))
```

```
i am living in the New York --> [('i', 'NN'), ('am', 'VBP'), ('living', 'VBG'), ('in', 'IN'), ('the', 'DT'), T
ree('GPE', [('New', 'NNP'), ('York', 'NNP')])]
```

```
My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERSON', [('Srikanth', 'NN
P'), ('Varma', 'NNP')])]
```

We did chunking for above two lines and then We got one list where each word is mapped to a POS(parts of speech) and also if you see "New York" and "Srikanth Varma",

they got combined and represented as a tree and "New York" was referred as "GPE" and "Srikanth Varma" was referred as "PERSON".

so now you have to Combine the "New York" with "_" i.e "New_York"

and remove the "Srikanth Varma" from the above sentence because it is a person.

13. Replace all the digits with space i.e delete all the digits.

> In the above sample document, the 6th line have digit 100, so we have to remove that.

14. After doing above points, we observed there might be few word's like

"_word_" (i.e starting and ending with the _), "_word" (i.e starting with the _),

"word_" (i.e ending with the _) remove the _ from these type of words.

15. We also observed some words like "OneLetter_word"- eg: d_berlin,

"TwoLetters_word" - eg: dr_berlin , in these words we remove the "OneLetter_" (d_berlin ==> berlin) and

"TwoLetters_" (de_berlin ==> berlin). i.e remove the words

which are length less than or equal to 2 after splitting those words by "_".

16. Convert all the words into lower case and lower case

and remove the words which are greater than or equal to 15 or less than or equal to 2.

17. replace all the words except "A-Za-z_" with space.

18. Now You got Preprocessed Text, email, subject. create a dataframe with those.

Below are the columns of the df.



In [0]: 1 data.columns

```
Index(['text', 'class', 'preprocessed_text', 'preprocessed_subject',  
      'preprocessed_emails'],  
      dtype='object')
```

In [0]:

```
1 data.iloc[400]
```

```
text          From: arc1@ukc.ac.uk (Tony Curtis)\r\r\r\r\nSubj...
class          alt.atheism
preprocessed_text  said re is article if followed the quoting rig...
preprocessed_subject  christian morality is
preprocessed_emails  ukc mac macalstr edu
Name: 567, dtype: object
```

To get above mentioned data frame --> Try to Write Total Preprocessing steps in One Function Named Preprocess as below.

In [0]:

```
1 def preprocess(Input_Text):
2     """Do all the Preprocessing as shown above and
3     return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that Text_data"""
4     return (list_of_preprocessed_emails,subject,text)
```

Code checking:

After Writing preprocess function. call that function with the input text of 'alt.atheism_49960' doc and print the output of the preprocess function

This will help us to evaluate faster, based on the output we can suggest you if there are any changes.

After writing Preprocess function, call the function for each of the document(18828 docs) and then create a dataframe as mentioned above.

Training The models to Classify:

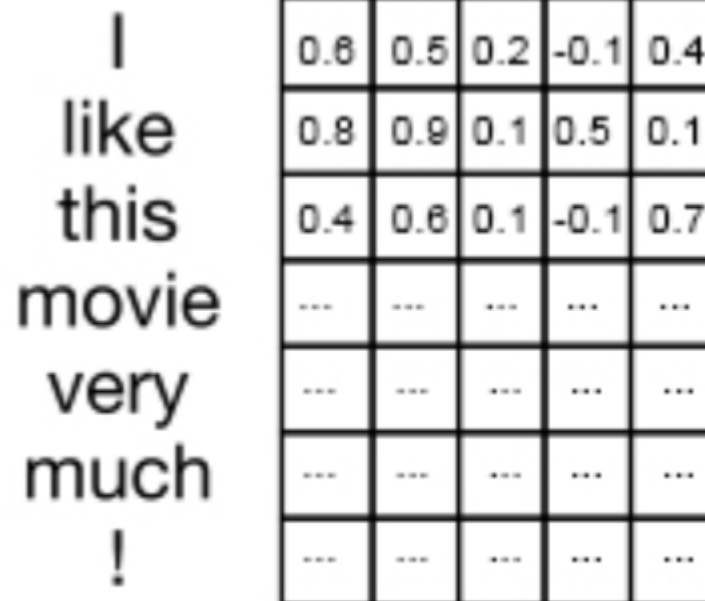
1. Combine "preprocessed_text", "preprocessed_subject", "preprocessed_emails" into one column. use that column to model.
2. Now Split the data into Train and test. use 25% for test also do a stratify split.
3. Analyze your text data and pad the sequence if required.
Sequence length is not restricted, you can use anything of your choice.
you need to give the reasoning
4. Do Tokenizer i.e convert text into numbers. please be careful while doing it.
if you are using tf.keras "Tokenizer" API, it removes the "_", but we need that.
5. code the model's (Model-1, Model-2) as discussed below
and try to optimize that models.
6. For every model use predefined Glove vectors.
Don't train any word vectors while Training the model.
7. Use "categorical_crossentropy" as Loss.
8. Use **Accuracy and Micro Averaged F1 score** as your as Key metrics to evaluate your model.
9. Use Tensorboard to plot the loss and Metrics based on the epoches.
10. Please save your best model weights in to 'best_model_L.h5' (L = 1 or 2).
11. You are free to choose any Activation function, learning rate, optimizer.
But have to use the same architecture which we are giving below.
12. You can add some layer to our architecture but you **deletion** of layer is not acceptable.
13. Try to use **Early Stopping** technique or any of the callback techniques that you did in the previous assignments.
14. For Every model save your model to image (Plot the model) with shapes
and include those images in the notebook markdown cell,
upload those images to Classroom. You can use "plot model"

please refer [this](https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model) (https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model) if you don't know now how to plot the model with shapes.

Model-1: Using 1D convolutions with word embeddings

Encoding of the Text --> For a given text data create a Matrix with Embedding layer as shown Below. In the example we have considered $d = 5$, but in this assignment we will get $d =$ dimension of Word vectors we are using.

i.e if we have maximum of 350 words in a sentence and embedding of 300 dim word vector, we result in 350×300 dimensional matrix for each sentence as output after embedding layer



The diagram illustrates the encoding of the sentence "I like this movie very much !" into a matrix of word embeddings. The sentence is shown on the left, and the resulting 7x5 matrix is shown on the right. The first three rows of the matrix contain numerical values, while the remaining four rows contain ellipses, indicating that the matrix is larger than shown.

I	0.6	0.5	0.2	-0.1	0.4
like	0.8	0.9	0.1	0.5	0.1
this	0.4	0.6	0.1	-0.1	0.7
movie
very
much
!

Ref: <https://i.imgur.com/kiVQuk1.png>

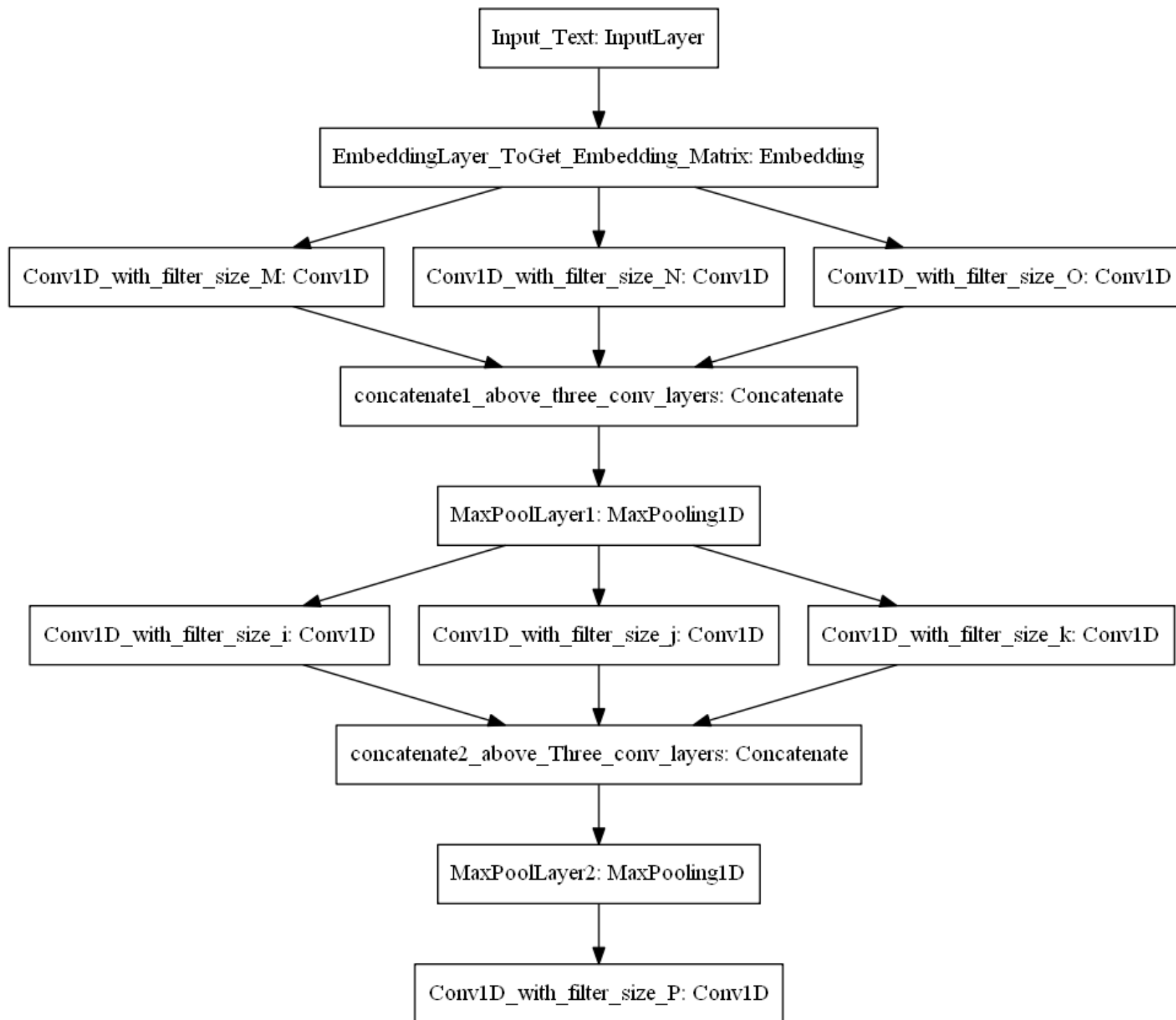
Reference:

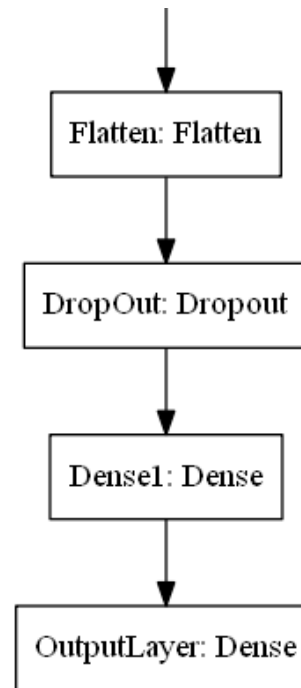
<https://stackoverflow.com/a/43399308/4084039> (<https://stackoverflow.com/a/43399308/4084039>).

<https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/> (<https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/>).

[How EMBEDDING LAYER WORKS](https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work) (<https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work>).

Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - <https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/> (<https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/>)





ref: '<https://i.imgur.com/fv1GvFJ.png>'

1. all are Conv1D layers with any number of filter and filter sizes, there is no restriction on this.
2. use concatenate layer is to concatenate all the filters/channels.
3. You can use any pool size and stride for maxpooling layer.
4. Don't use more than 16 filters in one Conv layer because it will increase the no of params.
(Only recommendation if you have less computing power)
5. You can use any number of layers after the Flatten Layer.

Model-2 : Using 1D convolutions with character embedding

Use 1D-convolutions!



Here are the some papers based on Char-CNN

1. Xiang Zhang, Junbo Zhao, Yann LeCun. [Character-level Convolutional Networks for Text Classification](http://arxiv.org/abs/1509.01626) (<http://arxiv.org/abs/1509.01626>). NIPS 2015
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. [Character-Aware Neural Language Models](https://arxiv.org/abs/1508.06615) (<https://arxiv.org/abs/1508.06615>). AAAI 2016
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. [An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling](https://arxiv.org/pdf/1803.01271.pdf) (<https://arxiv.org/pdf/1803.01271.pdf>).
4. Use the pretrained char embeddings <https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt> (<https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt>).

