Identification of grammatical gender of words in Hindi

Yash Malik (18075065)
Rhitwik Saha (18075048)
CSE, B.Tech. - 5th semester

Problem Statement

Determine the grammatical gender of words in isolation (not context aware), for Hindi language.

For example,

- जिम्मा [male]
- छेद [male]
- टेलीग्राम [male]
- प्रथा [female]
- साइंस [female]
- उन्नति [female]
- प्राचीन [any]
- सियासी [any]

Note

The words labelled with different genders at different places were removed.

That is, the context dependent words are not considered for training/evaluation/testing.

For example,

['डर', ['VM', 'any']]

['डर', ['NN', 'm']]

['मेरे', ['PRP', 'any']]

['मेरे', ['PRP', 'm']]

Approach

- ➔ **Binary Classification**
  Only consider words with a specific gender (male or female)

- ➔ **Complete Classification**
  Consider all words with context independent gender (‘male’, ‘female’ or ‘any’)

  - ◆ **Ternary Classification**
    Classify the words in three classes (‘male’, ‘female’, ‘any’)

  - ◆ **Multi-Label Classification**
    Classify the words in two classes (‘male’, ‘female’), with multiple labels possible

  - ◆ **Hybrid Approach**

Binary Classification
(the easier task)

Dataset (only words with ‘m’/‘f’ gender)

Training set:	87,033 words
Validation Set:	8,947 words
Unseen Test Set:	9,027 words

Solution Setup

चुनौती → च ु न ौ त ी → 5 10 31 16 28 2 → 5 10 31 16 28 2 0 0 0 0 → (neural network) → m / f

word → tokenization at character level → text vectorization → padding → neural network → output

Neural Network Structure

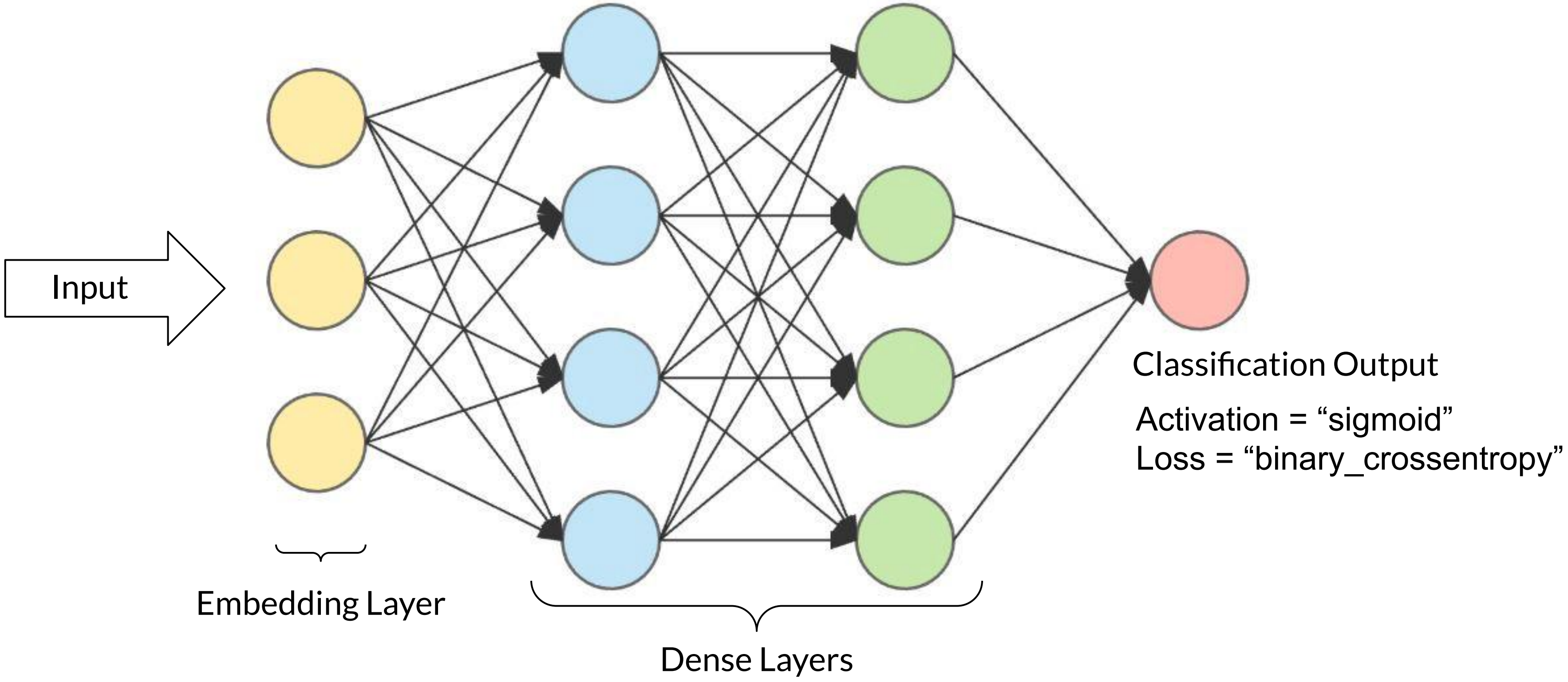


Image taken from https://www.kdnuggets.com/2019/02/neural-networks-intuition.html

Background Knowledge

- **Embedding Layer** : An embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.[1]
- **1D Global Average Pooling** : Used to replace the two dimensional tensor with a one dimensional tensor. [(input size) x (input channels)] ---> [input channels]

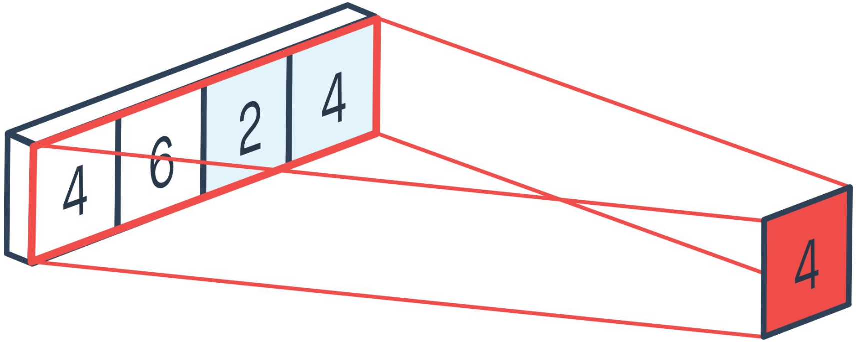


Image taken from https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/1d-global-average-pooling

1. https://www.tensorflow.org/tutorials/text/text_classification_rnn

Advantages of Global Average Pooling

"One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus the feature maps can be easily interpreted as categories confidence maps.

Another advantage is that there is no parameter to optimize in the global average pooling thus overfitting is avoided at this layer." [1]

1. Lin et al. in Network In Network (https://arxiv.org/pdf/1312.4400v3.pdf)

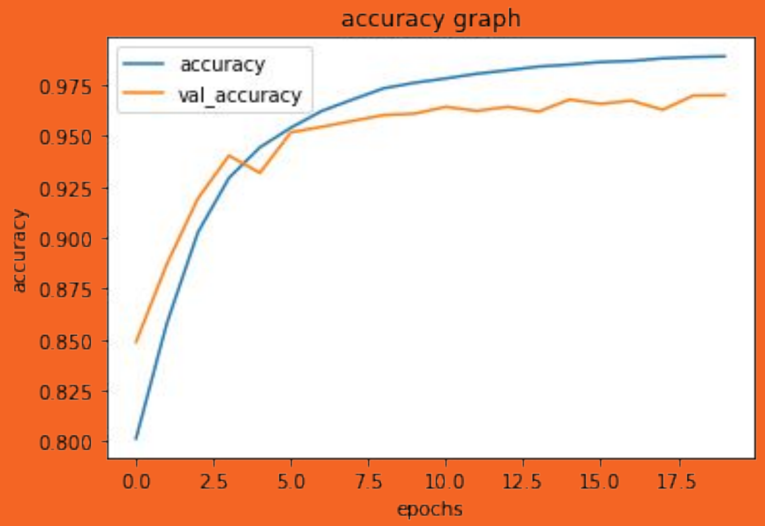
Model - 1

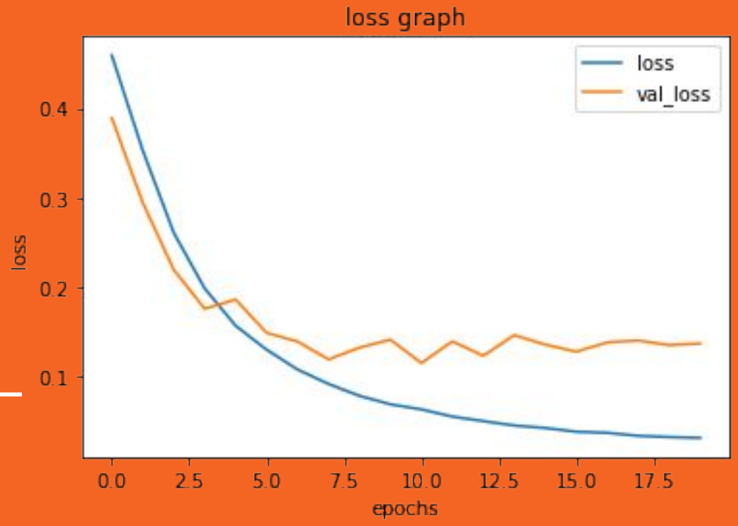
3 dense layers of size 512 each (can overfit the training data to 98.9% in 20 epochs)

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 20, 64)	5888
global_average_pooling1d (Gl	(None, 64)	0
dense (Dense)	(None, 512)	33280
dense_1 (Dense)	(None, 512)	262656
dense_2 (Dense)	(None, 512)	262656
dense_3 (Dense)	(None, 1)	513

Total params: 564,993
Trainable params: 564,993
Non-trainable params: 0





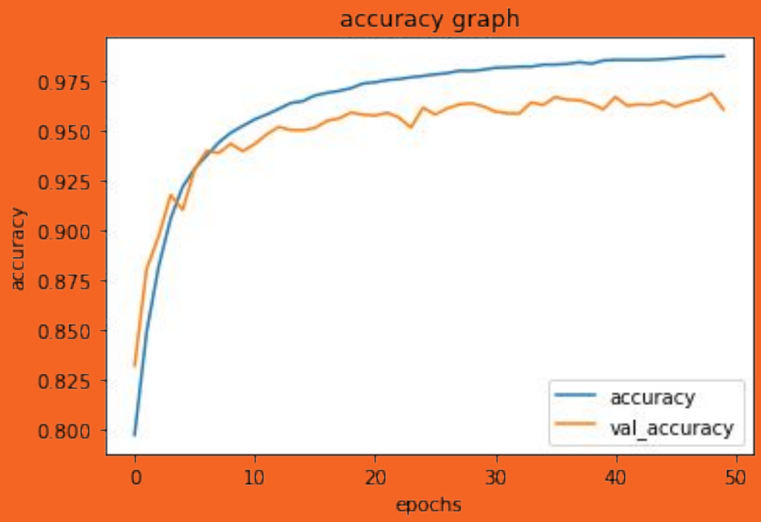
Model - 2

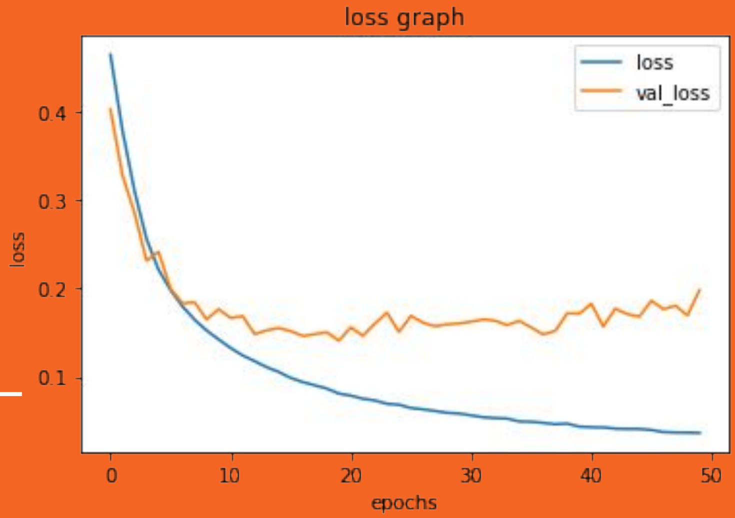
4 dense layers of size 64 each (can overfit the training data to 98.8% in 50 epochs)

Model: "sequential"

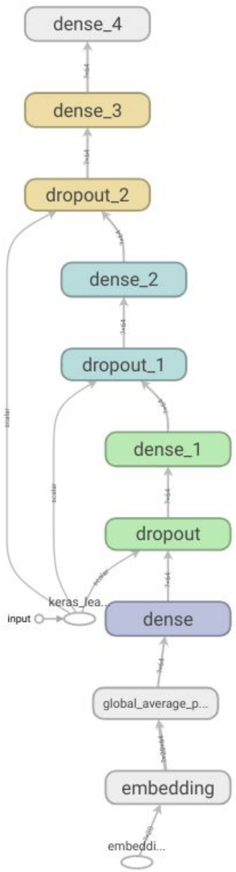
Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 20, 64)	5888
global_average_pooling1d (Gl	(None, 64)	0
dense (Dense)	(None, 64)	4160
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 1)	65

Total params: 22,593
Trainable params: 22,593
Non-trainable params: 0

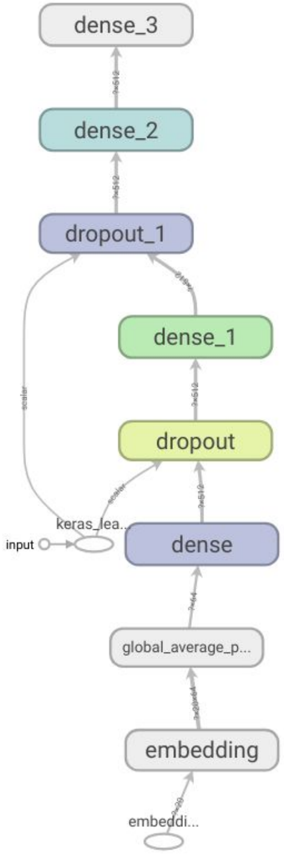




Final Models Used



1.



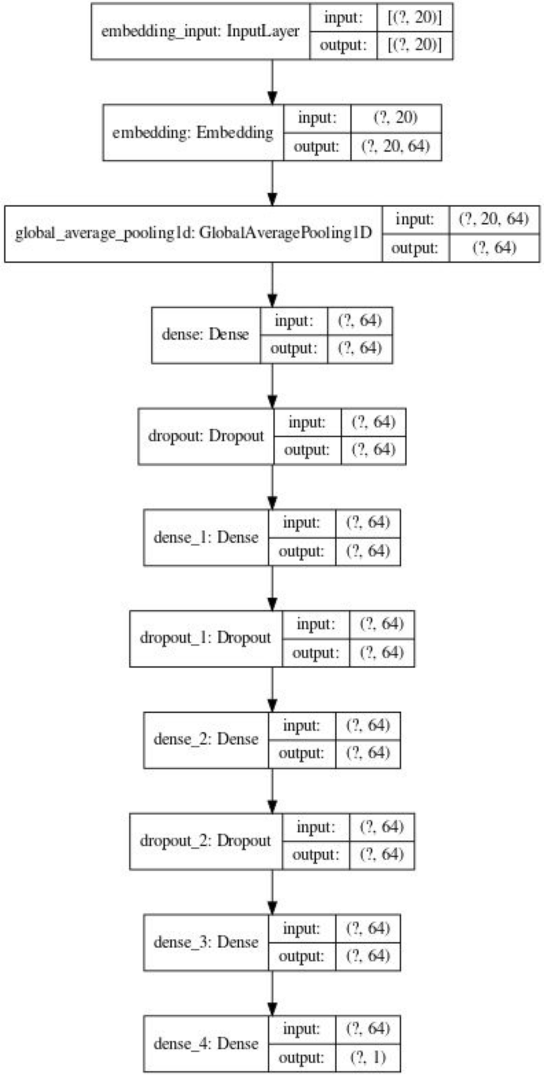
2.

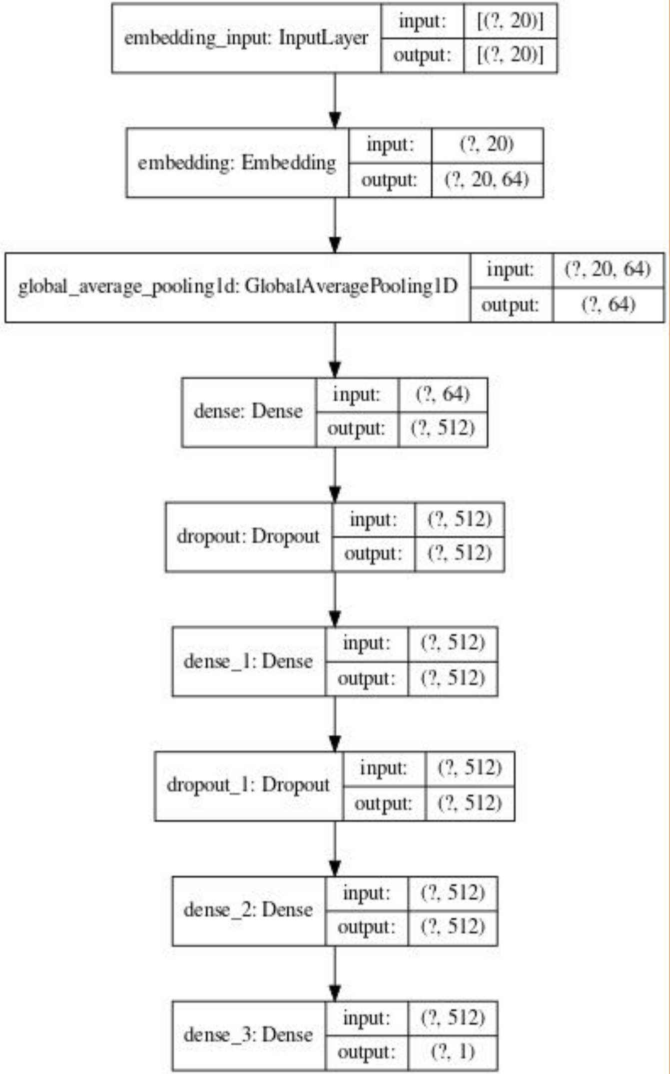
General Structure

- Embedding Input ->
- Embedding Layer ->
- Global Average Pooling 1D ->
- Dense + Dropout ->
- Dense ->
- Output Layer

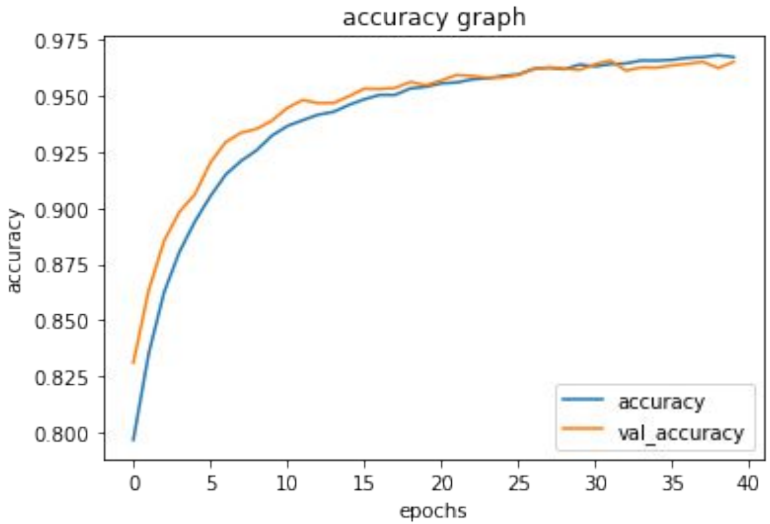
---

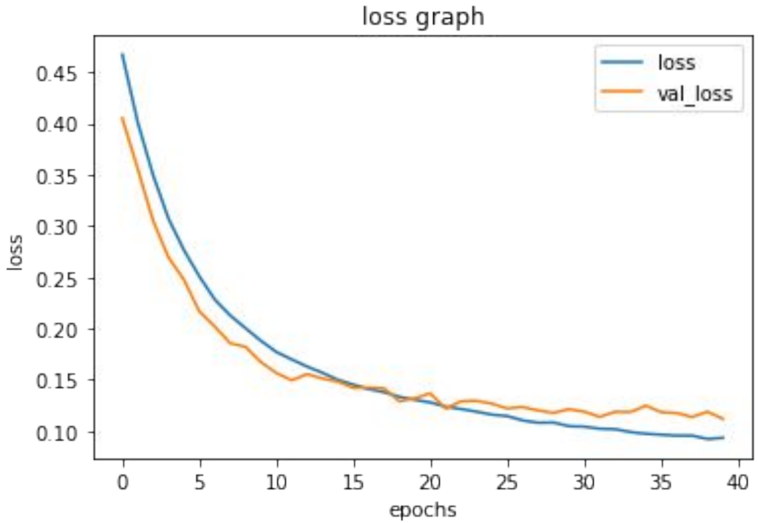
1. 4 dense layers of size 64 each, with dropout rate 0.2
2. 3 dense layers of size 512 each, with dropout rate 0.4





- Embedding size: **64**
- Maximum word size: **20**
- Activation Function for Dense Layers: **Relu**
- Activation Function for output layer: **Sigmoid**
- Loss function: **Binary Cross Entropy**



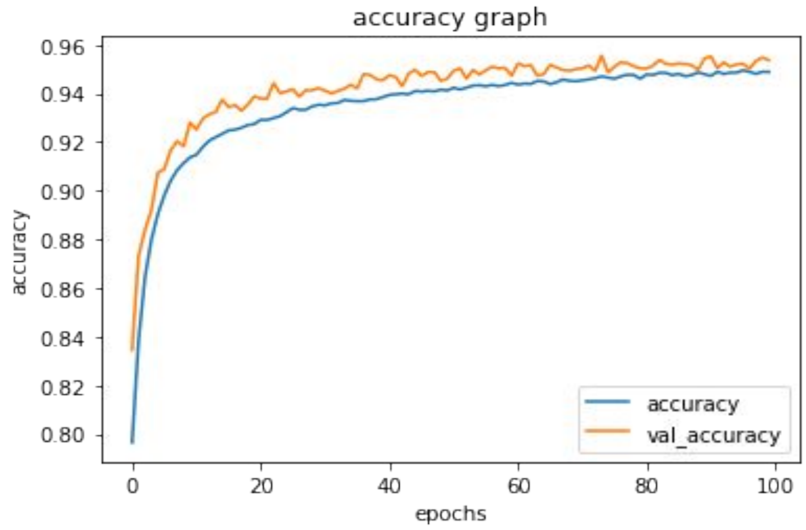


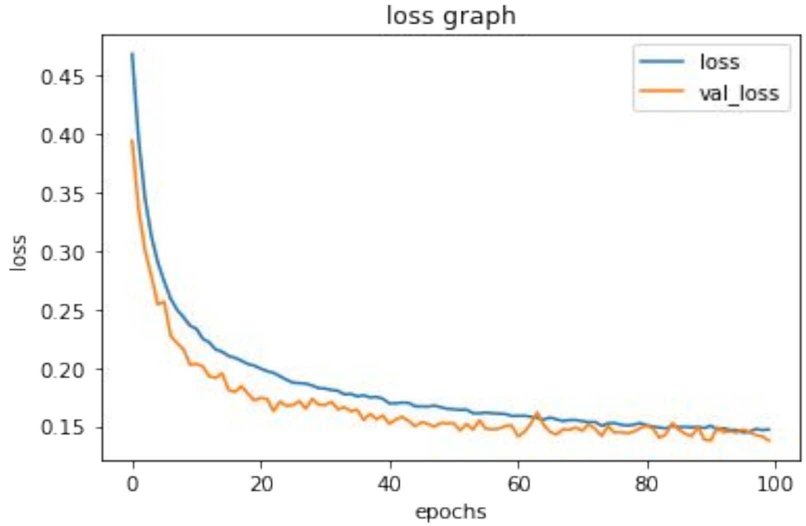
Best Results

for 3 dense layers of size 512 each with dropout (rate = 0.4) after the first 2 layers

Train Accuracy:
95.31% (20 epochs)
96.69% (40 epochs)

Validation Accuracy:
95.50% (20 epochs)
96.32% (40 epochs)





Best Results

for 4 dense layers of size 64 each with dropout (rate = 0.2) after the first 3 layers

Train Accuracy:
94.17% (50 epochs)
94.86% (100 epochs)

Validation Accuracy:
94.78% (50 epochs)
95.22% (100 epochs)

No significant gains are observed for more epochs. The accuracy slowly increases until the model starts to overfit.

Unseen Test Results:

Total Words = 9181

- **Model - 1**
  96.53 % accuracy
  m -> f errors = 146
  f -> m errors = 172

- **Model - 2**
  94.50 % accuracy
  m -> f errors = 242
  f -> m errors = 263

Analysis

A medium sized neural network was able to overfit the training set

➔ **Accuracy**

Both the models were able to perform quite well on the unseen test set

The larger model (Model - 1) performed, slightly better

➔ **Errors**

Apart from some valid errors, the errors mainly consisted of abbreviations or words imported from English language

Errors

- **Model - 1**

  जिम्मा [0.34234458]
  छेद [0.30132335]
  टेलीग्राम [0.00086647]

  प्रथा [0.9534087]
  साइंसेस [0.9872076]
  उन्नति [0.5486307]

- **Model - 2**

  पीटा [0.4883614]
  सिक्कों [0.02762693]
  खुलासे [0.44690883]

  धुन [0.9630592]
  नर्स [0.9736265]
  रैगिंग [0.5781765]

Complete Classification
(the actual task)

Dataset

Training set:	1,07,652 words
Validation Set:	10,845 words
Unseen Test Set:	11,137 words

Note

The words labelled with different genders at different places were already removed.

(3174 words)

Solution Setup

चुनौती → च ु न ौ त ी → 5 10 31 16 28 2 → 5 10 31 16 28 2 0 0 0 0 → neural network → m / f / any (ternary classification) Or m / f / m+f (multi-label classification)

word → tokenization at character level → text vectorization → padding → output

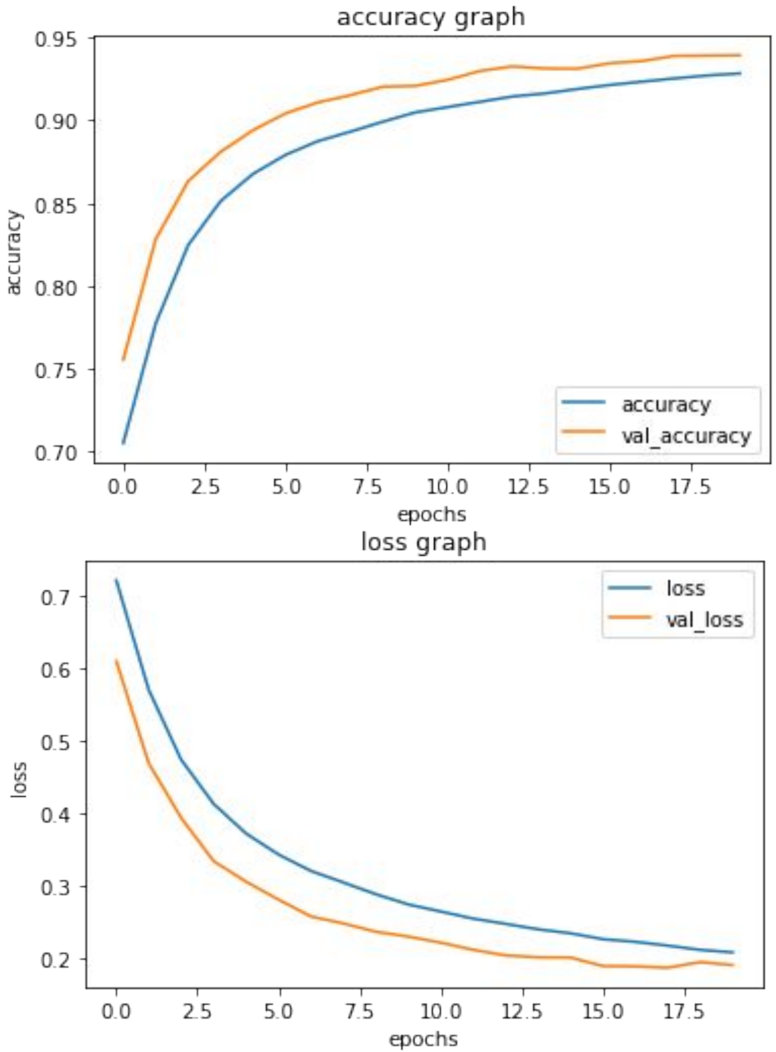
Ternary Classification

<u>Structure of Neural Networks Used</u>

Similar to the neural network used in Binary Classification, except the output layer:

1. **Three** neurons instead of *one* as output of neural network
2. Using "**softmax**" as the activation function, instead of "*sigmoid*"
3. Using "**categorical_crossentropy**" as the loss function, instead of "*binary_crossentropy*"

The output is a vector of size 3:
[1, 0, 0] or [0, 1, 0] or [0, 0, 1]

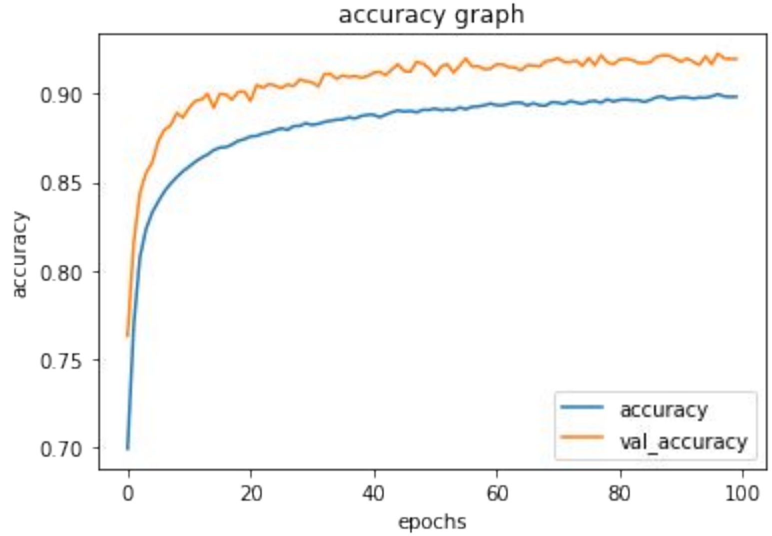


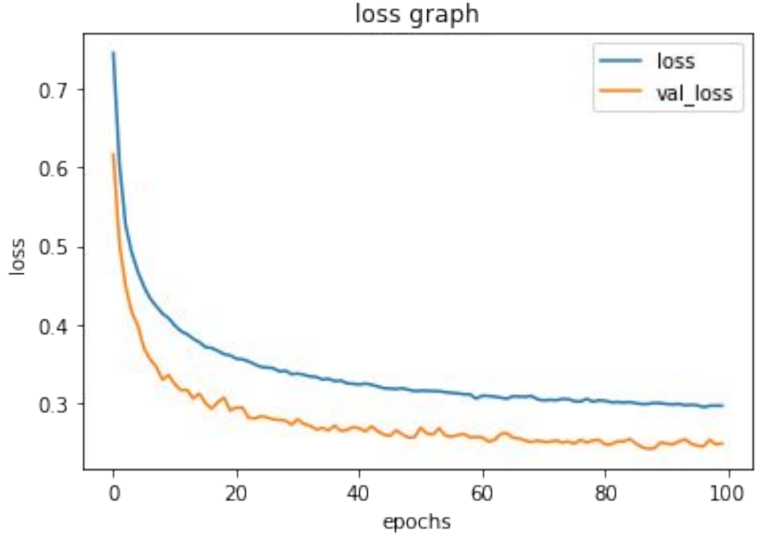
Results

for 3 dense layers of size 512 each with dropout (rate = 0.4) after the first 2 layers

Train Accuracy:
89.97% (10 epochs)
92.40% (20 epochs)

Validation Accuracy:
91.86% (10 epochs)
93.70% (20 epochs)





Results

for 4 dense layers of size 64 each with dropout (rate = 0.2) after the first 3 layers

Train Accuracy:
89.08% (50 epochs)
89.84% (100 epochs)

Validation Accuracy:
91.41% (50 epochs)
91.97% (100 epochs)

Unseen Test Results:

Total Words = 10968

- **Model - 1**
  93.71 % accuracy
  errors = 700

- **Model - 2**
  90.99 % accuracy
  errors = 988

Analysis

Both the models perform well, and the larger model outperforms the other

➔ **Accuracy**
The accuracy is still high though less than that of binary classification.

➔ **Why**
Accuracy slightly decreases but is still high, mainly because, majority of the words have specific gender

➔ **Errors**
The errors are mostly same, except the count increases due to the more complex nature of the problem.

Errors

- **Model - 1**

  मुठभेड़
  [0.349, 0.572, 0.077]
  [1, 0, 0]

  सियासी
  [0.903, 0.067, 0.028]
  [0, 0, 1]

- **Model - 2**

  सात
  [0.339, 0.495, 0.164]
  [0, 0, 1]

  प्राचीन
  [0.933, 0.051, 0.014]
  [0, 0, 1]

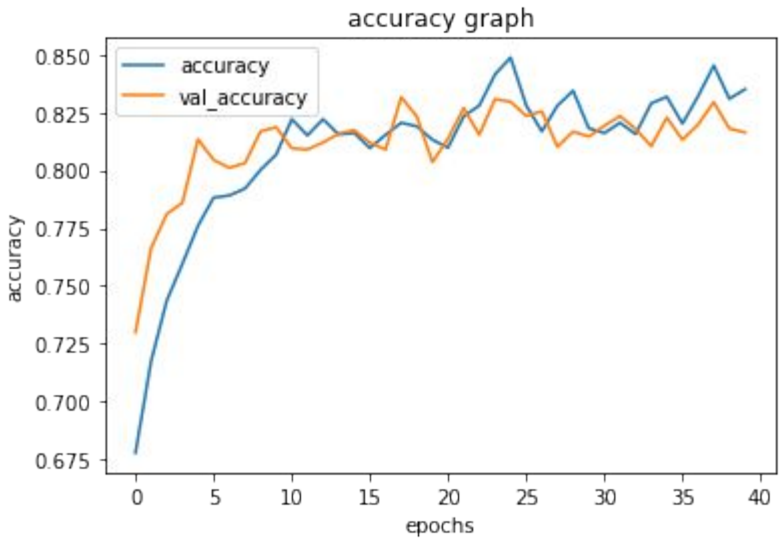
Multi-Label Classification

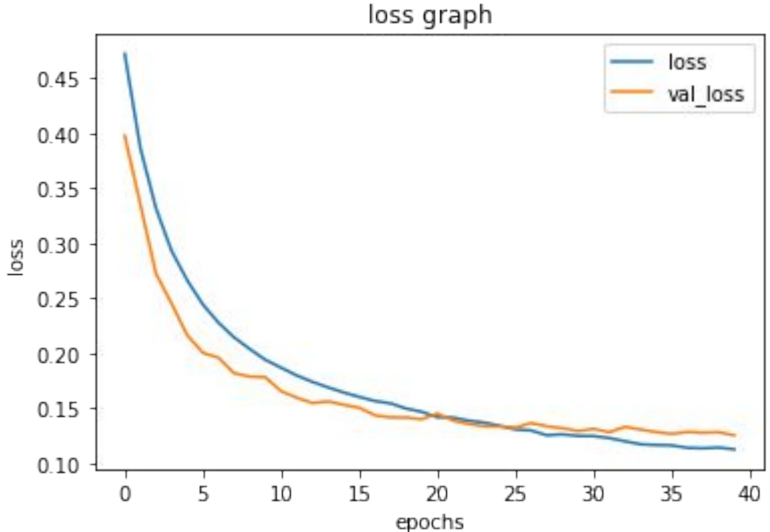
<u>Structure of Neural Networks Used</u>

Similar to the neural network used in Binary Classification, except the output layer:

1. **Two** neurons instead of *one* as output of neural network
2. Using "**softmax**" as the activation function, instead of "*sigmoid*"
3. Using "**binary_crossentropy**" as the loss function (same as in Binary Classification)

The output is a vector of size 2:
[*probability_female*, *probability_male*]





Best Results

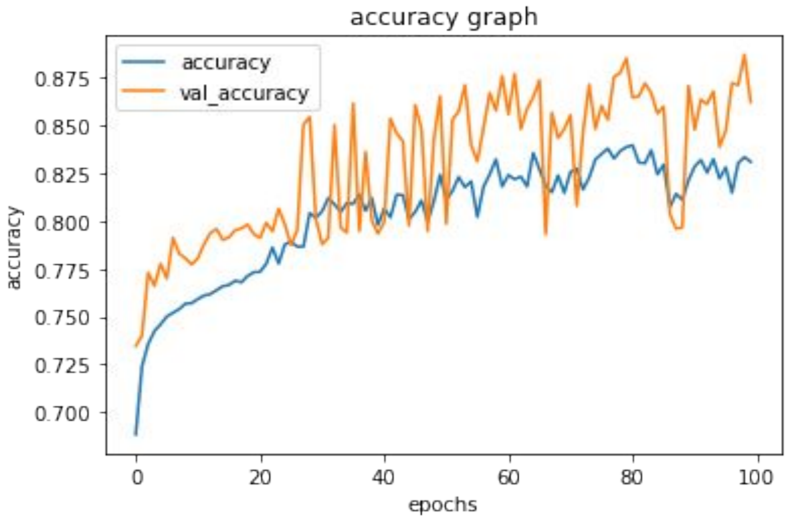
for 3 dense layers of size 512 each with dropout (rate = 0.4) after the first 2 layers

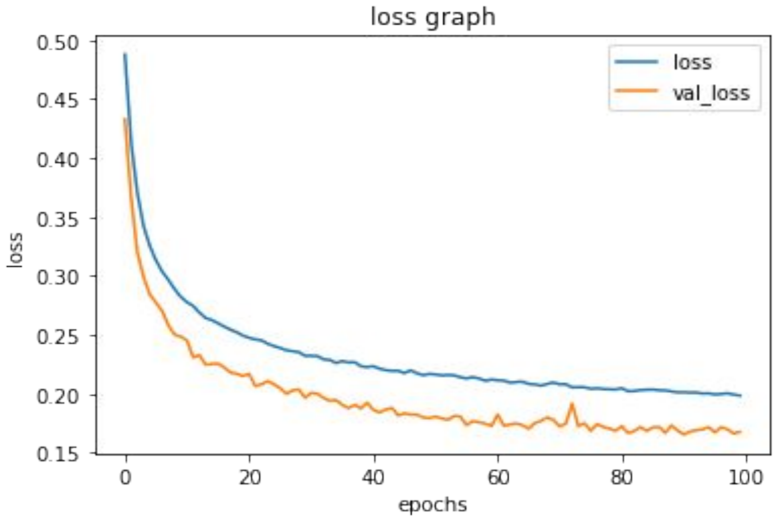
Train Accuracy:
81.34% (20 epochs)
83.38% (40 epochs)

Validation Accuracy:
81.39% (20 epochs)
81.89% (40 epochs)

Note

Even after allowing the model to overfit, the training accuracy remained close to 85%





Best Results

for 4 dense layers of size 64 each with dropout (rate = 0.2) after the first 3 layers

Train Accuracy:
81.40% (50 epochs)
82.96% (100 epochs)

Validation Accuracy:
83.69% (50 epochs)
86.87% (100 epochs)

Note

Even after allowing the model to overfit, the training accuracy remained close to 89%

Unseen Test Results:

Total Words = 11137

- **Model - 1**
  81.29 % accuracy
  errors = 681

- **Model - 2**
  85.12 % accuracy
  errors = 1060

Analysis

The smaller model shows higher accuracy on the test set

➔ **Accuracy**
The accuracy takes a serious hit, even though it was expected to perform slightly better

➔ **Why**
The count of errors is less, but the accuracy is still low. This can be attributed to the nature of model (loss function)

➔ Therefore, the model is not training in the best way

Errors

- **Model - 1**

  यूनीफॉर्म
  [0.290 0.818] [1, 0]

  आखिरी
  [0.419 0.933] [1, 1]

- **Model - 2**

  स्वस्थ
  [0.364 0.866] [1, 1]

  जुमलेबाजी
  [0.365 0.786] [1, 0]

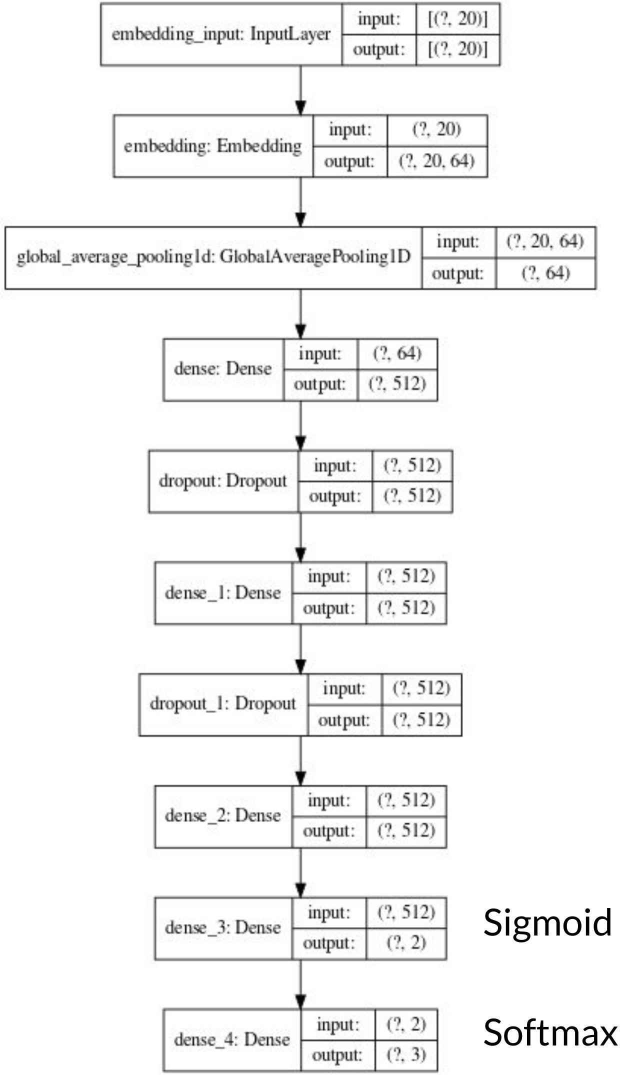
  द्रविड़
  [0.071 0.976] [1, 1]

Add another layer at the end of the model used for Multi Label Classification

(kind of assisting the ternary classifier)

[*probability_female*, *probability_male*] → [1, 0, 0] or [0, 1, 0] or [0, 0, 1]

Multi - Label + Multi - Class



Sigmoid

Softmax

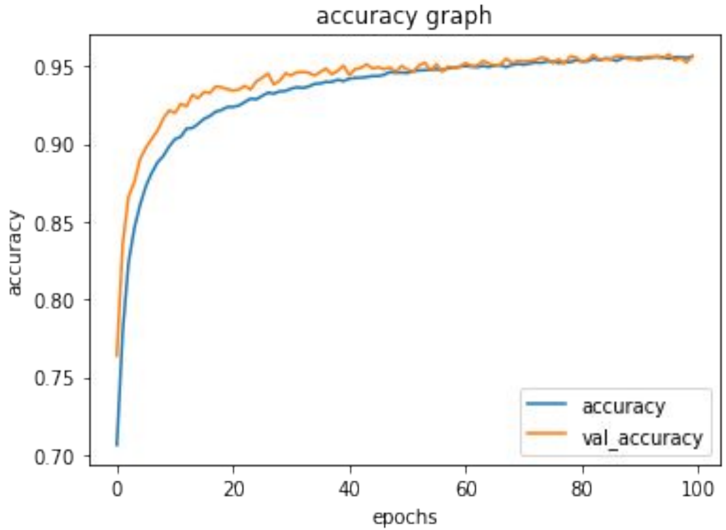
(Multi-Label + Multi-Class)

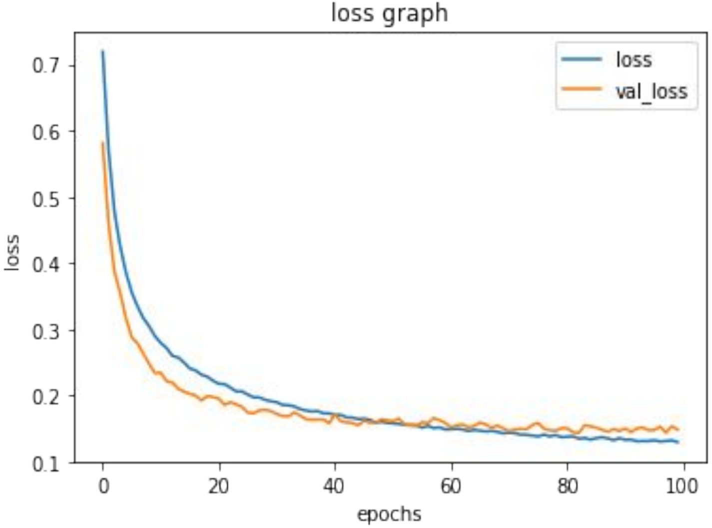
3 dense layers of size 512 each
(no overfitting observed till 100 epochs)

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 20, 64)	5888
global_average_pooling1d (Gl	(None, 64)	0
dense (Dense)	(None, 512)	33280
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
dense_3 (Dense)	(None, 2)	1026
dense_4 (Dense)	(None, 3)	9

Total params: 565,515
Trainable params: 565,515
Non-trainable params: 0





<u>Train Accuracy</u>:
93.99% (50 epochs)
95.13% (100 epochs)

<u>Validation Accuracy</u>:
94.75% (50 epochs)
95.52% (100 epochs)

<u>Unseen Test Results:</u>

Total Words = 11137

94.92 % accuracy
errors = 566

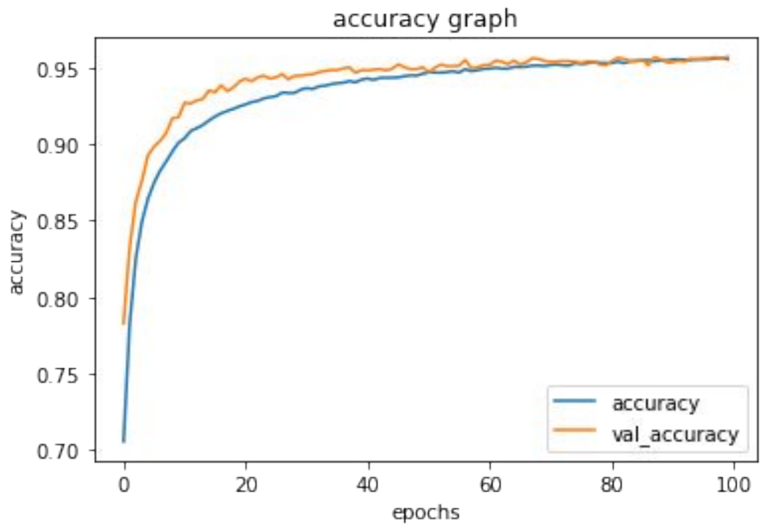
Note

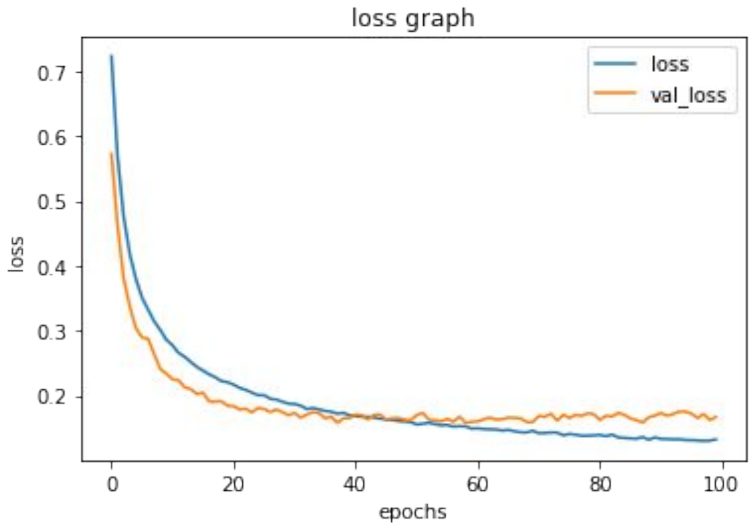
No regularization technique was used, as serious overfitting was not observed.

The accuracy now increases drastically, and the model performs quite well even on the full task

(at par with the results of the binary classification model).

What if **we allow the original ternary classification model** to figure this out on its own?





Train Accuracy:
94.58% (50 epochs)
95.59% (100 epochs)

Validation Accuracy:
94.86% (50 epochs)
95.60% (100 epochs)

Unseen Test Results:

Total Words = 11137

95.43 % accuracy
errors = 509

Note

No regularization technique was used, as serious overfitting was not observed.

Analysis

The ternary classification model on its own performs slightly better than the 'hybrid model'.

- **Accuracy**
  The accuracy is quite high for both the hybrid and ternary classification.
- **Why**
  It is expected that the ternary model figures out the (male + female = any) property on its own.

Errors

- सीएमटीडीयू
  [0.1248, 0.8746, 0.0005]
  [1, 0, 0]
- हिमलिंग
  [0.7867, 0.1403, 0.0728]
  [0, 1, 0]
- 467
  [0.0, 0.0, 1.0]
  [0, 1, 0]
- बेअसर
  [0.0035, 0.9918, 0.0046]
  [0, 0, 1]
- काली
  [0.1364, 0.8188, 0.0447]
  [1, 0, 0]

Summary

Binary Classification

Test Accuracy: 96.53 %

Multi-Label Classification

Test Accuracy: 81.29 %
(even though the count of errors was less)

Subtask

Full Task

Ternary Classification

Test Accuracy: 93.7%

Hybrid/Ternary (with 100 epochs)

Test Accuracy: 94.92/95.43%

Possible Future Work

To observe the accuracy trends according to the word categories

(proper noun, auxiliary verbs, abbreviations, ...)

Try RNN/CNN based models

To take the context in account. This can help in determining the gender for words which were earlier excluded

(might also improve accuracy?)