1 Question 1

With a true label 1, the log loss is reduced to $-log(pi)$. For a confident correct prediction ($y_i = 1$) the log loss is 0. For a strongly incorrect prediction the loss goes to infinity. As we can see in the Figure 1 the loss grows exponentially with respect to the difference between the true label and the predicted one. The more the prediction is far from the true value the more the penalization.

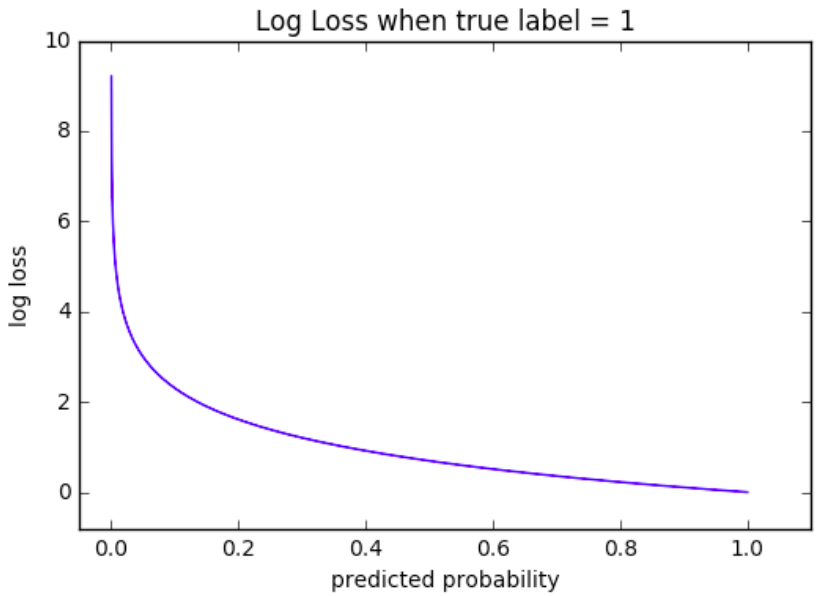

Figure 1: log loss with true label 1.

2 Question 2

The value is -3. It is the sum of the row wise dot product of the filter W_0 by the second slided window of the text or the 2nd region matrix $A[2:5]$ plus the bias b_0. The result is

$$W_0.A[2:5] + b_0 = -3$$

3 Question 3

Since it is a binary classification we can have only 1 unit in the final layer and applying a sigmoid activation $\sigma(x) = \frac{1}{1+exp(-x)}$ to it will give us a value in $[0, 1]$ that is considered as a probability of belonging to one of the two classes that are chosen with regard a threshold and we use it to compute the loss. This will help us reduce the number of parameters.

4 Question 4

We have:
d : dimensionality of the word embedding space
V : Vocabulary size
n_f : number of filters
h : size of filters
- The Word embeddings are updated during training so the number of parameters is the size of the embedding matrix $\rightarrow$ nb params$= (V + 2) \times d$ (+2 for the padding and out-of-vocabulary (OOV) rows)

- One branch CNN → nb params $= n_f \times (h \times d + 1)$: We considered a bias per filter
- In maxpooling and dropout there are no trainable parameters
- Last Dense layer to perform binary classification → nb params $= n_f \times 1 + 1$ (1 for bias)

$$totalparams = (V + 2) \times d + n_f \times (h \times d + 2) + 1$$

5 Question 5

Figures 2 and 3 prove that our model is learning a meaningful internal document representations. We see that after training the model is constructing a document embedding space that tend to separate between the two classes.

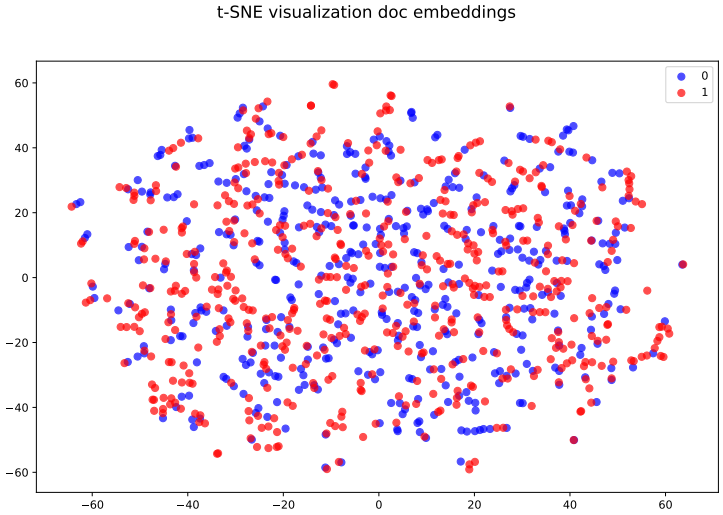

Figure 2: document internal representation before training.

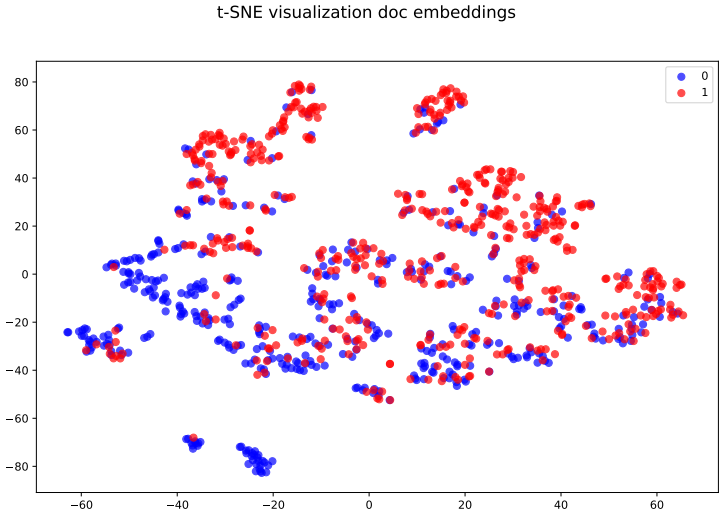

Figure 3: document internal representation after training.

6 Question 6

After calculating the saliency map of the text
'Oh , god , this was such a disappointment ! Worst movie ever . Not worth the 15 bucks .'
as shown in Figure 4 We can interpret that the word $'worst$ has the most effect on the prediction of this

example, since it have the biggest magnitude so a small change in its embedding vector would affect the most the prediction. This also shows that our model is considering the right words like '!' and '$ever$' and it ignores non necessary words for classification like '$this$' and 'was'.

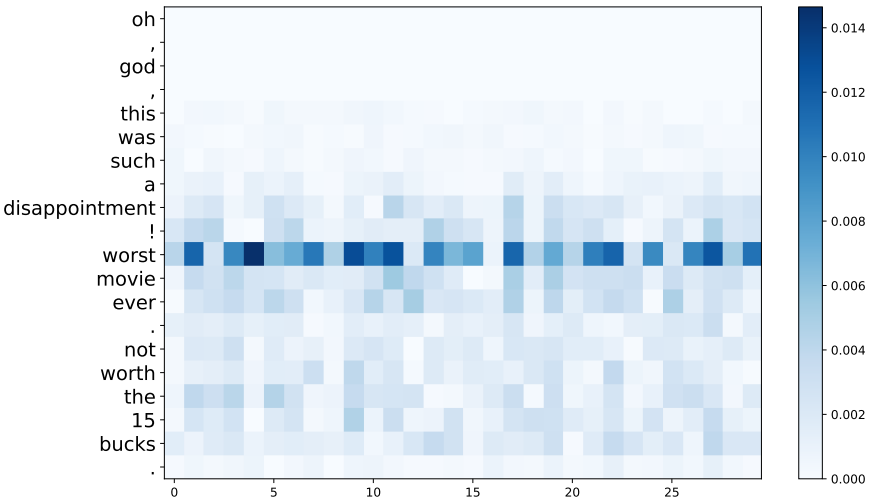

Figure 4: Saliency map.

7 Question 7

CNN's are good at extracting local and position-invariant features but it does not capture long range semantic dependency of words. It just consider local key-phrases. So when the result is determined by the entire sentence or a long-range semantic dependency CNN is not effective as shown here [1].
Even if there is the option to use filters with larger sizes, this does not resolve the problem since it considers only full pattern and not just part of it. In other words, if there are unnecessary words in the middle that will change from sentence to sentence it can't ignore them.
Since not all words contribute equally to the representation of the sentence meaning, we want a mechanism that give different weights to words of the sentence which is relative to its contribution in the meaning. One solution to this are Attention models.

References

[1] Mo Yu Wenpeng Yin, Katharina Kann and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. In *arXiv preprint arXiv:1702.01923*, 2017.