Text2image

1. Introduction:

Traditionally, Natural Language Processing (NLP) has been considered more difficult than image recognition, at least in the case of Deep Learning. Hence, Image recognition models are generally faster to train, require comparatively smaller amount of data and tend to achieve higher accuracy. Therefore, a new method is proposed for representing text in the form of an image. Apart from the new approach, other improvements including mapping values according to their position, plotting values on log scale, gaussian filtering are proposed. All code can for this can be found here.

1.1 Inspiration

The idea of representing other forms of data in form of images, was first seen in fraud detection using mouse movement [1]. The coordinates of its movement were plotted as a colour image, showing various factors like speed, direction, acceleration etc.

1.2 The data

The dataset used for all experiments is a subset of the Fake News dataset by George Mclntire. It contains approximately 1000 articles of fake and real news, combined.

2. Feature Extraction

2.1 Pre-Processing

The data is lowercased, all special characters removed and the text and title concatenated. Words appearing in more than 85% of the document are also removed. Additionally, a list of words is explicitly avoided (stopwords). The one used is a standard list of stop-words, mostly uninformative repeating words. Modifying the stopwords for classification of fake news can be an area to explore in the future, especially to bring out writing style particular to fake news

2.2 Calculating TF-IDF

For scoring and extracting the keywords, a scikit-learn implementation of a ***smooth*** *term frequency – inverse document (tf-idf)* is used. The IDF is calculated separately for both the fake and real news corpus. Calculating separate IDF scores resulted in a massive jump in accuracy in comparison with a single IDF score for the entire corpus.

,

The TF-IDF scores are then calculated iteratively for each document. Here, the title and the text are not scored separately, but together.

3. Generating heatmaps

3.1 Processing TF-IDF values

For each document, 49 of the words with the highest TF-IDF values are extracted. These words are then used to create a 7x7 array. Here, the amount of words chosen can act like a hyper-parameter. For shorter, simpler pieces of text fewer words can be used while a larger amount of words can be utilized to represent longer, more complex text. For this dataset, 49 seems to be a good size.

Instead of arranging the TF-IDF values in descending order of their magnitude, they are mapped in accordance to their position in the text. The TF-IDF values are mapped in this way as it is seen more representative of the text and provides richer features for the model to train on. Since a single word can appear multiple times in a piece of text, it's first occurrence is taken into account.

3.2 Plotting the heatmaps

3.2.1 Logarithmic scale

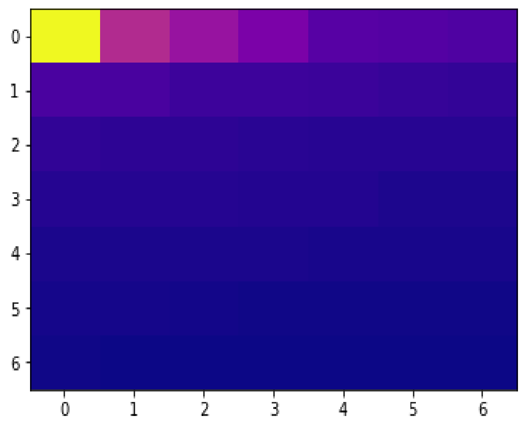
Instead of plotting the TF-IDF values as is, all values are plotted in log scale. This is done, to reduce the large difference between the top and bottom values

Highest Values	Lowest Values
0.578	0.052
0.361	0.048
0.248	0.047
0.197	0.037

Table 1. Difference between the top 4 TF-IDF values v/s the bottom 4

While plotting, because of this difference, majority of the heatmap will not show any colour variance. Hence, they are plotted on log scale to better bring out the differences.

,



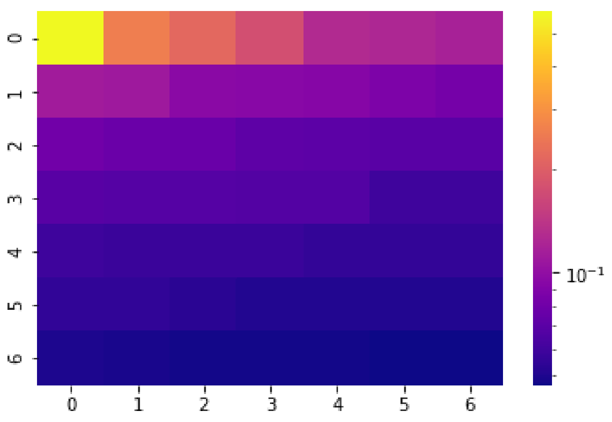


Figure 1 (left) shows the TF-IDF values plotted as is. Figure 2 (right) shows the same values plotted on log scale

3.2.2 Mapping values

Earlier versions, took only the first occurrence of a word and plotted the values. Hence, the plots did not give the whole picture. Later, the values were plotted as they were and repeated occurrences were allowed. Since, we can have one word appear more than once in a plot, larger heatmaps of size 12 x 12 were possible. Earlier, these were not possible due to the limited number of unique words. Mapping values resulted in a ~ 6 % increase in accuracy. This can be explained as the heatmaps are now more representative.

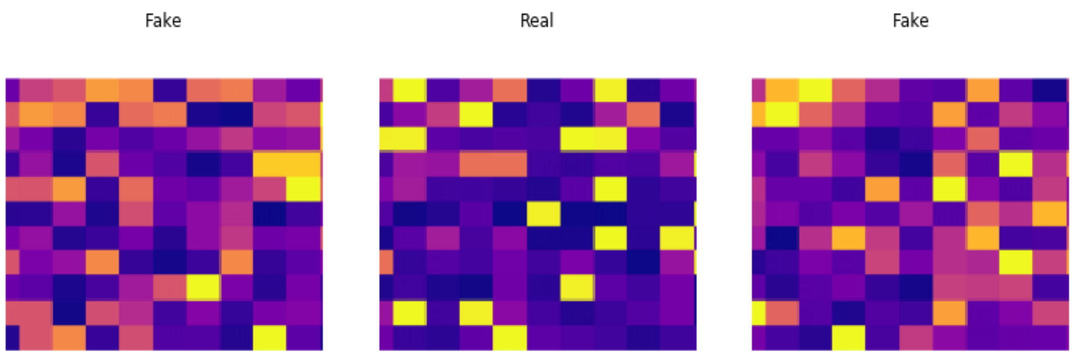
3.2.3 Gaussian filtering

One of the downfalls is the large amount of overfitting while training the model. This can be attributed to the lack of any data augmentation as currently, no way of data augmentation seems viable for this use case. Hence, gaussian filtering was used on the entire dataset in a bid to smoothen out the plots. While it did decrease the accuracy by a little, there was a significant decrease in overfitting especially during initial stages of training.

3.3 Final plots

The final heatmaps are of the dimension 11 x 11 and are plotted with seaborn. As, both the x-axis and the y-axis as well as the colour bar do not give us any information while training, we remove them. The type of heatmap used is ‘plasma’ as it showed quite good colour variation. Experimenting with different colour combinations can be an area to explore in the future. Here is an example of how the final plots look

,



4. Training the model

The model is trained on a resnet34 using fast.ai. 489 fake articles and 511 real articles were used in total.A standard 80:20 split was used between train and test set with no data augmentation. All code used can be found here.

5. Results.

No. Of epcohs	Training Loss	Validation Loss	Accuracy
0	0.934111	0.930879	0.522843
1	0.811310	0.928518	0.553299
2	0.675762	0.665015	0.639594
3	0.591624	0.642079	0.675127
4	0.451970	0.635090	0.690355
5	0.379864	0.704963	0.609137
6	0.316208	0.762437	0.675127
7	0.243443	0.640104	0.680203
8	0.186887	0.620586	0.700508

6. Conclusion

After 9 epcohs an accuracy of 70.05% was achieved. The model does overfit a lot suggesting the need of developing data augmentation techniques for this approach.

,

6.1 Further experiments

Although just the beginning, this approach shows promising signs. One of the major

problems to be addressed is the large overfitting and data augmentation techniques. Several methods including interchanging values in the array, randomly removing values (similar to dropout) can be tried. It was also observed that changing the colour palette of the plot had a significant difference in the accuracy achieved. Further study of the same is required to ascertain the best possible colour scheme.

References

[1]https://www.splunk.com/blog/2017/04/18/deep-learning-with-splunk-and-tensorflow-for-security-catching-the-fraudster-in-neural-networks-with-behavioral-biometrics.html

,