COSC 6368 – AI

FINAL PROJECT – SUMMER 2019

https://github.com/bdalal/COSC6368AI-FinalProject

GENDER CLASSIFICATION OF BLOG AUTHORS

BINOY DALAL (1794070)

ABSTRACT

This paper comprises the report for the final project for COSC 6342 - Artificial Intelligence course offered at the University of Houston in Summer of 2019. In the final project I attempt to build and improve on my previous work on this topic from my COSC 6342 - Machine Learning course that I had enrolled for in Spring of 2019. Gender classification of authors based on their written work has been a useful area of research from the perspectives of authorship attribution and verification, digital forensics as well as online marketing. Here I attempt to leverage a different deep learning approach compared to what I had used in my original attempt for COSC 6342 and attempt to beat the baselines I established in that project.

INTRODUCTION

Gender classification is an important problem for a multitude of reasons. Knowing the gender of the author of a piece of text can play a big role in determining the owners of polarizing ideas and divisive social forces which have a direct impact on the security and well being of any society. On a different but equally serious note, gender classification of text is also important to determine the rightful owners of intellectual property in disputes. It is also a useful tool in directed marketing campaigns when business owners want to promote gender specific products. Knowing what brands and/or products different genders prefer can greatly help a business increase sales and simultaneously deliver greater customer satisfaction. For the purposes of this project, I'll be assuming only two genders - male and female and will be focusing on classification on the basis of blogs. A blog (a truncation of "weblog") is a discussion or informational website published on the World Wide Web consisting of discrete, often informal diary-style text entries (posts) [1].

Classification of blogs is a difficult task because the text is very informal. It consists of slang, swear words, emoticons, spelling errors, and misplaced punctuation and incorrect grammar. Defining any sort of structure in the text is therefore a challenging task. This lack of structure directly impacts the ability of a learning algorithm to generalize well over the text. One can however view these very problems as features on which to train the learning algorithm – things like the no. of typos made, emoticons used etc. by different genders.

SURVEY

The topic of gender classification based on blogs and other text has been extensively worked upon in the past.

[2] and [3] have explored the notion of measuring a text's implicitness and develops a unitary measure of a text's relative contextuality (implicitness), as opposed to its formality (explicitness). They define a score called F-Measure which indicates contextuality vs formality. Lower scores indicate contextuality while higher scores indicate formality. The F-Measure is then defined as:

$F = 0.5 * [(nounfrq + adjfrq + prepfrq + artfrq]) - (pronfrq + verbfrq + advfrq + intfrq) + 100]$

Here we consider the frequencies of nouns, adjectives, prepositions, articles, pronouns, verbs, and adverbs. They have found the F-Measure to be a good differentiating factor between text written by males and females. The F-Measure for men generally tends to be on the higher side when compared to females on the current blog corpus.

Stylistic features have been analyzed in depth by [4] and have been found to have a tremendous differentiating capability between genders. Stylistic features capture the writing styles of authors and can include words, characters and POS (Parts-of-Speech) as well as sequences thereof.

[5] and [6] have considered the use of gender preferential features, namely certain kinds of words that are preferred by one gender over the other. They have found that women use more emotionally descriptive words like 'terribly', 'awfully' etc. while men express more independence in their conversational patterns. They consider words ending with 'able', 'al', 'ful', 'ible', 'ic', 'ive', 'less', 'ly', 'ous' and words conveying apologetic meaning like sorry,a pology etc.

Certain categories of words can also serve to act as discriminating features as noted by [4] and [7]. These categories referred to as word factors are used to capture meaning in text. Each word factor represents some theme like Home, Family, Business etc. It has been found that the genders have a greater inclination towards writing about a certain subset of these word factors. More than 20 word factors are considered between [4] and [7] to be used as features. Some of them can be found in Table 1.

Additionally, in my own work for this project for my COSC 6342 class, I've used numerical characteristics like the number of proper nouns, upper case words, sentence lengths and others[1].

1 For a complete list, the reader is referred to the implementation of the project at https://github.com/bdalal/COSC6342ML_Spring19_Project

I had found certain features to showcase a huge difference between the two genders. Table 2 lists out these numbers. All the numbers have been averaged across the entire corpus of data. [8]

Factors	Words
Family	years, family, mother, children, father, kids, parents, old, year, child, son, married, sister, dad, brother, moved, age, young, months, three, wife, living, college, four, high, five, died, six, baby, boy, spend, christmas
Work	work, working, job, trying, right, met, figure, meet, start, better, starting, try, worked, idea
Home	woke, home, sleep, today, eat, tired, wake, watch, watched, dinner, ate, bed, day, house, tv, early, boring, yesterday, watching, sit
Business	system, based, process, business, control, example, personal, experience, general
Positive	absolutely, abundance, ace, active, admirable, adore,agree, amazing, appealing, attraction, bargain, beaming, beautiful, best, better, boost, breakthrough, breeze,brilliant, brimming, charming, clean, clear, colorful,compliment, confidence, cool, courteous, cuddly, dazzling, delicious, delightful, dynamic, easy, ecstatic, efficient, enhance, enjoy, enormous, excellent, exotic,expert, exquisite, flair, free, generous, genius, great,graceful, heavenly, ideal, immaculate, impressive, incredible, inspire, luxurious, outstanding, royal, speed,splendid, spectacular, superb, sweet, sure, supreme,terrific, treat, treasure, ultra, unbeatable, ultimate,unique, wow, zest

Table 1. Some Word Factors

Characteristic	Males	Females
Character length	2406.11	2259.61
Proper noun counts	40.53	32.81
Average sentence length	132.57	113.19
Upper case characters	73.96	66.46
Title case words	53.45	48.85

Table 2. Numerical characteristics

All of the above are hand engineered features and they have been shown to work well in [2], [3], [4], [5] and [7]. I had however tried a different approach of using automatically extracted features[8]. This was primarily done in two ways – the first way was to feed the text directly to a Bidirectional LSTM preceded by an Embedding layer which generated vector representations for the tokens. The second approach was to use Doc2Vec[9] to generate document vectors which were then fed to traditional classification models. I achieved decent results but they were nowhere near the state of the art set by [4]. The best accuracy I could get on the test set and in cross validation was around 68% with the Doc2Vec approach and around 59% with the RNN model.

There are also certain freely/commercially available gender classifiers online like GenderGenie[10] and uClassify[11] that attempt to determine the gender based on supplied text. These systems also deliver moderately decent results often misclassifying a lot of blogs.

[12] has implemented a promising deep learning based approach which uses windowed recurrent - convolutional neural networks based on the design proposed in [13] and have shown to get accuracy on blogs comparable to the state-of-the-art set by [4].

[4] has also curated a corpus of blogs crawled from online sources which will be used in my experiments for this project. They have compiled roughly 3200 blogs and manually verified the genders of authors to make sure the data has no noise and is clean.

MY IMPLEMENTATION

DATA

For my experiments I will be using the blog corpus from [4]. This corpus has 3212 blogs in total. 1671 are written by males and 1541 are written by females.

There's a slight class imbalance here in favor of the males though not by too much and as such shouldn't impose a lot of bias on the final classification results. A comparative study of the impact of this class imbalance is shown later.

The data is split into train, validation and test sets in an 80-10-10 proportion respectively, i.e. 80% of the data is used for training, 10% is used for validation during training and the remaining 10% is used for testing.

SETUP

All experiments are carried out on my personal laptop and a cloud computing instance on Google Cloud. The specs are as follows:

Laptop: 2.2 GHz 2-core processor, 16 GB RAM, Nvidia GeForce GTX 940MX with 2GB VRAM

Cloud instance: 2.3 GHz 4-core processor, 16GB RAM, Nvidia Tesla P4 with 8 GB VRAM.

I've used the following frameworks to implement my deep learning model: Keras with Tensorflow backend on Python 3.5.3.

I'm also using pre-trained word embeddings specifically the glove-twitter-100d embeddings from [14].

MODEL

My model is based loosely off of the implementation proposed in [12]. The main idea presented by [12] is to use the left and right contexts for every word in the text. The left context is defined by all the words preceding the current word and the right context is defined by all the words following the current words. Both the contexts use a fixed window size which can be considered a hyperparameter for tuning the model. The embeddings for each word in the 2 contexts (left and right) and the actual word are then concatenated together and a linear transformation is applied with a non-linear tanh activation. The vectors generated by this layer are passed through a max-pooling layer which takes the element-wise max over the output. This means that the ith element in the output of this layer is the maximum of the ith

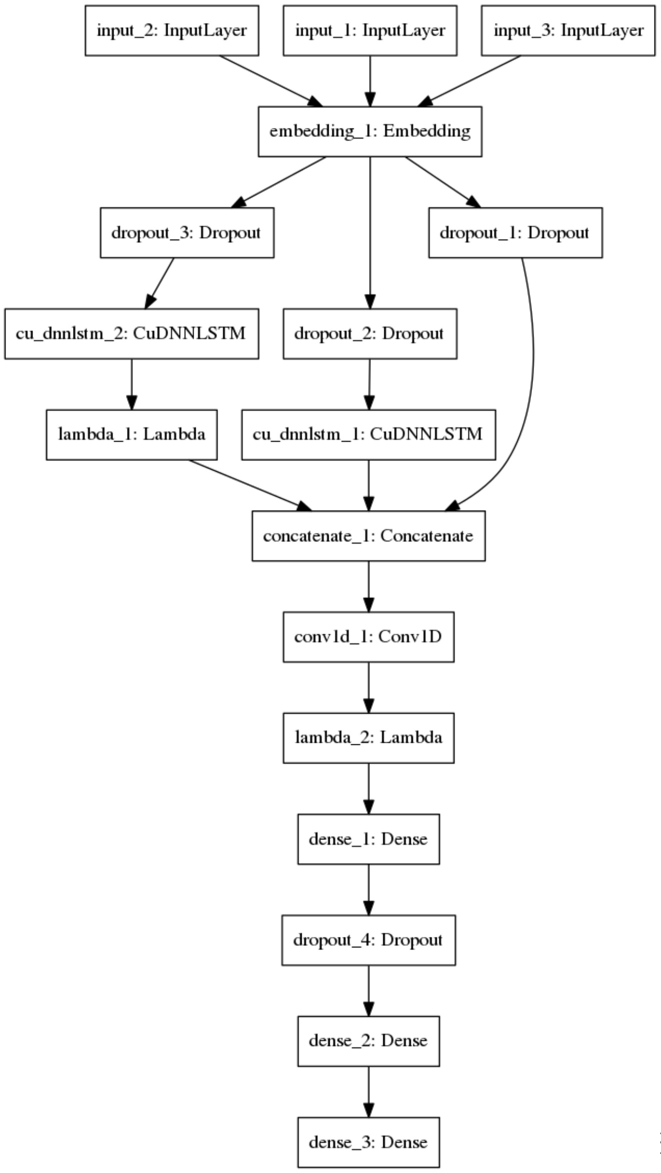

Illustration 1: My WRCNN model

elements in all the vectors of the input. The final layer is a dense layer with a simple linear activation followed by softmax for classification over multiple classes. For more details on the architecture along with the mathematical background, the reader is referred to [12]. I've omitted the equations and the architecture diagram here for brevity.

In my approach, I'm using a pre-trained embedding layer with weights loaded from glove-twitter-100d embeddings as mentioned before. The reason I chose these embeddings is because twitter users normally use a lot of slang and the tweets themselves are often grammatically incorrect and have typos, emoticons etc. It is therefore representative of the dataset at hand. The embedding layer is however set to be trained through backprop. The 2 contexts and the actual tokens are generated and passed through this embedding layer to get the vectors. All the 3 embeddings are subject to dropout layers with the dropout rate set to 0.1 which means that 10% of the units in the layer are randomly dropped out during training. The contexts are then passed through an LSTM layer with the right context being reversed after output from the LSTM layer. All these outputs are then concatenated together and passed through a 1D convolutional layer with filter size = 100 and kernel size = 1. This is similar to applying a TimeDistributed Dense layer at each point of the output. Max-pooling is then applied and the vectors are passed through 3 dense layers with Dropout added in between followed by sigmoid activation. Since there are only two classes, I've used a sigmoid activation instead of softmax, since using softmax in keras introduced more complexity and made the evaluation of results more confusing for me.

I used binary crossentropy as the loss function which is defined as *loss = (target) * log(sigmoid(output)) + (1 – target) * log(1 – sigmoid(output))*.

EXPERIMENTS

Most of my time in this project was devoted towards reducing overfitting on the corpus by trying out different approaches like using regularization and dropout at the different layers in conjunction with trying out different number/types of layers in the same architecture. Depending on the no. of trainable parameters, the no. of epochs the model is trained for and whether I used my laptop or the cloud instance, the training times have varied anywhere from 5 – 90 minutes per run.

The experiments have been tabulated in Table 3. along with train/validation accuracy and loss charts on the model from Figure 1 shown in Figures 2 and 3 respectively.

For all tests, I've considered a maximum possible 500 words with the left and right contexts comprising of 199 words each. The hardware I have couldn't support bigger sequences. Padding for shorter pieces of text was applied to the end of the document. I found through manual inspection of some blogs that more authors explicitly mentioned their gender through their writing in the first few lines as compared to the later portions of the blogs.

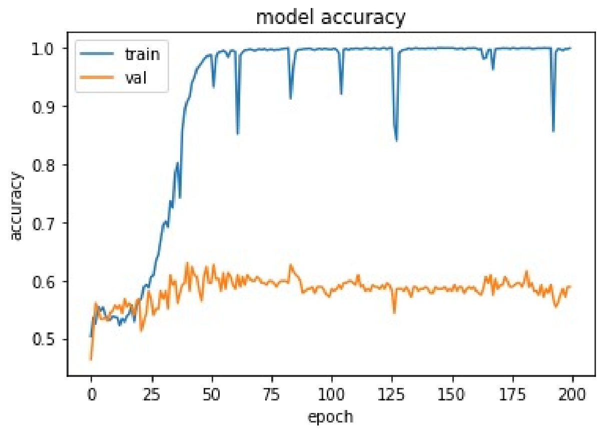

Figure 2. Train/Val accuracy

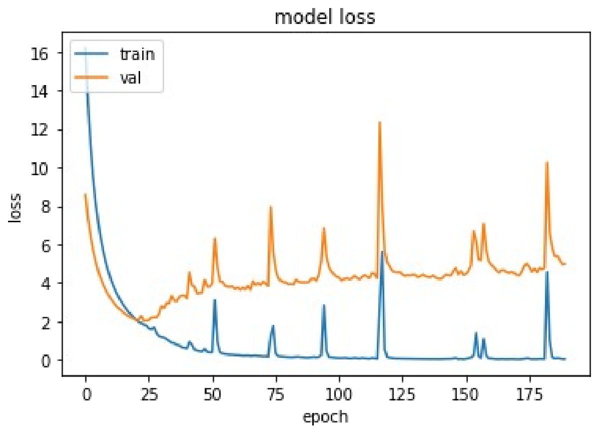

Figure 3. Train/Val loss

No. of trainable params	Difference in Model from Illustration 1	No. of epochs	Training Loss / Accuracy at final epoch	Validation Loss / Accuracy at final epoch
119,949,877	N / A (Model in Illustration 1)	200	0.0522 / 0.9996	4.9875 / 0.5882
553,477	Bidirectional LSTM for all embeddings with embeddings frozen	200	0.2805 / 0.9842	1.6390 / 0.5848
119,904,977	Bidirectional LSTM for all embeddings	46	0.6824 / 0.9948	5.9585 / 0.5744

Table 3. Results

From the above results we can see that none of the architectures generalize well, but all of them have a tendency to overfit, which means they do understand the data well. Therefore, increased generalization through regularization and dropout will help and difference combinations will need to be explored. We can also see that model with Bidirectional embeddings for all layers converges to near perfect accuracy on train data quite quickly compared to the other architectures probably because the bidirectional layers preserve and provide a lot more context for the model to start approximating the data well fairly quickly.

Table 4 shows a comparison on the model from Illustration 1 when the classes are balanced vs. the original imbalanced configuration. All reported figures are from the last training epoch.

Balanced			**Imbalanced**		
Train loss / accuracy	**Validation loss / accuracy**	**Test loss / accuracy**	**Train loss / accuracy**	**Validation loss / accuracy**	**Test loss / accuracy**
0.0426 / 0.9992	4.8029 / 0.5827	4.6291 / 0.589	0.0522 / 0.9996	4.9875 / 0.5882	4.6208 / 0.5497

Table 4. Comparison between performance on balanced vs imbalanced corpus

We can see for the balanced dataset that while the train and validation accuracies are pretty similar, the test results are much better because both the classes are equally represented therefore removing any class related bias we may have had earlier.

Besides the above approach I also earlier tried a different approach where I used the POS tags extracted from the corpus to train a glove word embedding model. My rationale behind this was that since previous literature has identified that writing styles can be captured using POS tags, by training an embedding model on them, I could establish relationships between different POS tags which in turn would capture the stylistic features. This did not work out very well as the accuracy was poor (~52% on the entire dataset after training for 10 epochs) and I therefore abandoned the idea. I’d used POS tags extracted during my COSC 6342 project and GloVe to train a model on these.

CONCLUSION

Given the results, we can clearly make out that the model is overfitting on the training data by a large margin. There are a few ways to overcome this – apply dropout and regularization or use different architectures. I've tried both dropout and regularization but the sheer number of combinations of these layers have prevented me from a coming up with better tuning for my model where in starts to deliver better validation accuracy. More experimentation is required to come up with better hyperparameters and combinations for these layers to enable better generalization.

My main goal for this project was to try out the WRCNN architecture on the corpus which I didn't get a chance to try during COSC 6342. In that sense I've achieved the goal for this project but the results are by no means promising. My key takeaway from this project is that coming up with good architectures to solve deep learning problems is quite hard and even when references in the form of research publications are available, it is still challenging to implement a published architecture correctly especially when no code is available publicly. I did get a much better understanding of how to apply Keras and the functionalities it provides to create different deep learning models. This was also my first experience in implementing a model from a published paper which was a good learning experience as well.

Another approach that I tried out partially was to use [16]'s autoencoder implementation to encode the blogs to a compressed representation and the pass them through a traditional classifier. This failed because the model wasn't able to achieve low enough loss on the dataset, since it couldn't handle more than 500 tokens per blog given my hardware setup. In the future I may try out this approach further on better underlying hardware to enable better training of the autoencoder model.

From the above, we can see that gender classification based on blogs is a hard task mainly owing to the informal nature of the content. There are however ways to achieve significant results on this data with better implementations and more sophisticated architectures.

REFERENCES

[1] Wikipedia contributors. Blog — Wikipedia, the free encyclopedia, 2019. [Online; accessed 24-July-2019]

[2] Francis Heylighen and Jean-Marc Dewaele. 2002. Variation in the contextuality of language: An empirical measure. Foundations of Science, 7(3):293–340.

[3] Scott Nowson, Jon Oberlander, Alastair J Gill, et al. 2005. Weblogs, genres and individual differences. In Proceedings of the 27th Annual Conference of the Cognitive Science Society, volume 1666, page 1671. Stresa.

[4] Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In EMNLP.

[5] Malcolm Corney, Olivier De Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In 18th Annual Computer Security Applications Conference, 2002. Proceedings., pages 282–289. IEEE.

[6] Bibliography of gender and language. http://ccat.sas.upenn.edu/haroldfs/op-cult/bibliogs/gender/genbib.htm.

[7] Shlomo Argamon, Moshe Koppel, James Pennebaker, and Jonathan Schler. 2007a. Mining the blogosphere: Age, gender and the varieties of self-expression. First Monday, 12.

[8] Binoy Dalal 2019. Gender Classification of Blog Authors. Class project for COSC 6342 – Machine Learning (Spring term). https://github.com/bdalal/COSC6342ML_Spring19_Project

[9] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In International conference on machine learning, pages 1188–1196.

[10] http://www.hackerfactor.com/GenderGuesser.php#Analyze

[11] https://www.uclassify.com/browse/uclassify/genderanalyzer_v5

[12] Aric Bartle and Jim Zheng. 2015. Gender classification with deep learning. In Technical report. The Stanford NLP Group.

[13] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao. Recurrent Convolutional Neural Networks for Text Classification. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015

[14] https://github.com/RaRe-Technologies/gensim-data

[15] https://github.com/airalcorn2/Recurrent-Convolutional-Neural-Network-Text-Classifier

[16] https://github.com/erickrf/autoencoder