Gender Classification of Blog Authors

Binoy Dalal
University of Houston
brdalal@uh.edu

Abstract

This paper comprises of the report component of the course project for COSC 6342 - Machine Learning offered at the University of Houston for the Spring 2019 term. Automatic classification of genders of blog authors has many applications ranging from those in the commercial domain (targeted marketing) to forensics where it may be imperative to know the gender of an author in order to investigate a case. This paper attempts to beat the baselines set by (Mukherjee and Liu, 2010) to further improve the classification of genders of blog authors while also showcasing my learnings in the process of working on this project. For the purposes of this report we'll assume the existence of two genders only - male and female.

1 Introduction

A blog, short for weblog, is a discussion or informational website published on the World Wide Web consisting of discrete, often informal diary-style text entries (posts). Classifying the gender of blog authors can lend utility to many tasks. It could benefit commercial applications by enabling companies to understand the preferences of males and females with respect to their products by automatically analyzing blogs about their products and classifying the writings by gender. Additionally, such a classification system could also be used to analyze blogs that spread misinformation and hateful or polarizing ideologies that could result in civil disorder or violent acts. By classifying the genders of the authors of such blogs, state authorities may be able to curtail the damage cause by such incendiary writings.

Classifying blogs by gender is a difficult tasks because of their informal nature. Authors tend to use slang, emoticons, extra punctuations as well as tend to introduce spelling errors (typos), often on purpose. Such irregularities can pose problems for any machine learning system to learn the necessary features needed to classify these blogs successfully. However, such irregularities can often be discriminating features when it comes to analyzing genders because they reflect the inherent differences in the natures of males and females. In addition to such features, other classes of features like the usage of POS tags, content words, numerical characteristics and feature selection is also common in related literature (Schler et al., 2006) and (Argamon et al., 2007b).

My method uses all of the above ideas to try and build a discriminating gender classifier for blog authors.

2 Features

I consider two sets of features in my attempt. The first set is engineered features which have been explicitly extracted from text and are then processed and passed to the classifier. The other features are those extracted by deeplearning methods - RNN and Doc2Vec (Le and Mikolov, 2014).

2.1 Engineered Features

The features below comprise of hand-engineered features extracted from the corpus using text analysis methods.

2.1.1 F-Measure

(Heylighen and Dewaele, 2002) and (Nowson et al., 2005) explore 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:

$$F = 0.5 * [(nounfrq + adjfrq + prepfrq + artfrq]) - (pronfrq + verbfrq + advfrq + intfrq) + 100] \quad (1)$$

Here, we consider the frequencies of nouns, adjectives, prepositions, articles, pronouns, verbs, adverbs and prepositions.

2.1.2 Stylistic Features

Stylistic features capture writing styles of authors. These can be represented by the words, characters or underlying POS tags of the blogs. In this attempt, I consider all three - the words, the POS tags and the characters. I evaluate use of individual words, as well as word, character and POS n-grams in conjunction.

2.1.3 Gender Preferential Features

Gender preferential features are a set of words that can serve as helpful discriminators between different genders. Women tend to use more emotionally descriptive words like "terribly", "awfully" etc. Men on the other hand, express independence in their conversational patterns. This feature choice is based on research performed by (Corney et al., 2002) and (sch). In my attempt I consider the counts of these features throughout the corpus.

Feature	Words
f1	words ending with able
f2	words ending with al
f3	words ending with ful
f4	words ending with ible
f5	words ending with ic
f6	words ending with ive
f7	words ending with less
f8	words ending with ly
f9	words ending with ous
f10	sorry words

Table 1: Gender Preferential Features

2.1.4 Word Factors

(Argamon et al., 2007b) and (Mukherjee and Liu, 2010) suggest the use of certain categories of words as discriminating features. These categories are also called word factors and they serve to capture meaning in the text. Each factor represents a general theme like Home, Family, etc. and also positivem negative and neutral emotions. A set of words is present for each factor. I consider the counts of these words towards the factor. For a complete list of words, the reader is referred to (Argamon et al., 2007b) and (Mukherjee and Liu, 2010). All words are compared against the corpus in a case insensitive manner.

2.1.5 Numerical characteristics

Numerical characteristics like the number of proper nouns or upper case words amongst others can also shed light on important discriminating features between men and women. The table below shows certain features that show a big difference between the blogs of men and women. All numbers have been averaged across the entire corpus. In addition to the features above, I also consider the number of punctuation, stop words and sentences.

Feature	Men	Women
Character Length	2406.11	2259.61
Proper Noun count	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

2.2 Automatic Feature Extraction

Deep learning methods are able to automatically extract features from text and invalidate the need to explicitly extract features. I've evaluated two methods of text representations using deep learning. The first is directly feeding the text to a recurrent neural network (RNN) and classifying it. The second approach is to tokenize text and generate document vectors using Doc2Vec (Le and Mikolov, 2014). The vectors are then fed as features into a traditional classifier. The details of the actual models are covered in the next section.

3 Methodology

3.1 Environment and Data

I've built my implementation using Python 3.6, Scikit-Learn (Pedregosa et al., 2011), Flair (Akbik et al., 2018), Gensim (Řehůřek and Sojka, 2010), XGBoost (Chen and Guestrin, 2016), Keras (Chollet et al., 2015) and Tensorflow (Abadi

et al., 2015). I'm using scikit-learn to perform preprocessing and feature extraction from text. This mainly concerns all the engineered features desrcibed in Section 2.1. The numerical features have all been scaled to a range between 0 and 1 to prevent unnecessary bias from bigger numbers for any one of the features. This also includes the F-Measure, gender preferential features and word factors. Keras and Tensorflow have been used to build the RNN while Gensim is used to build the Doc2Vec model. All the models have been built and trained on my personal laptop which has a 2.2 GHz dual-core processor, 16GB of memory and Nvidia GTX 940M graphics chipset with 256 Cuda cores and 2GB VRAM. I'm using the dataset from (Mukherjee and Liu, 2010) for the purposes of this project. The dataset has 3212 blogs in total with 1671 written by men and 1541 written by women. Since there is a clear imbalance in the classes, while training I'm randomly sampling a subset of the blogs written by males, which is the same as the number of blogs written by females, to offset this bias. I've split my data using a 70/20/10 train/val/test split. I performed 10-fold cross validation on the train/val split and used this to fine-tune the hyperparameters on my model. The final model was then evaluated on the test split.

3.2 Training Using Engineered Features

For the engineered features, I've evaluated the performance of my model with different n-gram ranges spanning from 1-gram tokens to 2-8 gram tokens. Additionally I have also considered the term frequencies as well as boolean feature values for this n-grams. POS tag extraction has been performed using Flair which is a state-of-the-art NLP parser and tagger. I was earlier using NLTK to generate POS tags but I found that the accuracy of NLTK tags is pretty poor. It was consistently labeling prepositions and other tags as proper nouns. The pattern I observed here was that if a sentence started with a particular word, and by extension had the first letter capitalized, NLTK would label it as a proper noun. After switching to Flair, I saw significant improvement in POS-tags being generated. The only downside to using Flair is that, since it uses a neural network to generate the tags, hardware constraints can cause significant performance issues in generating the tags. On my computer it took more than three hours to generate the tags on the entire corpus.

I used the χ^2 feature selection mechanism to choose the best features which I then used to train the classifier. I tried using mutual information gain as well, but it took too long to transform the data and also gave poorer results when compared to χ^2 selection.

I've evaluated the performance of two different classification models on the corpus. The first model is a multi-layer perceptron from scikit-learn which is basically a dense neural network. I have seen the best results with a network with 75 layers with 75 units each using the identity activation function and the Adam optimizer (Kingma and Ba, 2014). I have also implemented the XGBoost model which is a gradient boosted model (using decision tree classifiers). For this model I found that a max depth of 35, with 200 estimators provided the best results on the CV tests.

3.3 Training Using Doc2Vec

The Doc2Vec approach basically converts the input words into a vector representation but along with this also generates paragraph vectors which are built over sequences of words which can help capture long distance relationships between pieces of text. There are two models to build paragraph and word vector representations of a text. The first is called Paragraph Vector - Distributed Memory (PV-DM) which considers the order of words while generating the vectors. The second is called a Distributed Bag-of-Words (DBOW) model which considers the text as a bag-of-words and ignores the order of words. While it may seem counter-intuitive, my experiments show that the DBOW model performs better than the PV-DM model given the same set of parameters. I've used a window size of 5 for the model. This means that the model considers the 5 adjacent words to predicting the next tokens.

These vectors (features) were then fed as input to an MLP and an SVM classifier. The MLP configuration was the same as that used for then engineered features. The SVM configuration was standard with the penalty parameter set to 2500. Smaller values yielded worse results because the classifier was more constrained and was therefore not able to approximate the data well.

The main motivation behind using Doc2Vec is capture implicit features of the data which I cannot engineer by hand. I also wanted to learn about

more advanced techniques in machine learning which led me to experiment with this approach.

3.4 Training using RNN

The idea behind using a RNN is similar to Doc2Vec in that the RNN can possibly pick the best features without having to explicitly engineer them. An additional motivation was my desire to learn more about neural networks and how to use them in practice. RNNs are good for learning application related to text because they are able to establish relationships and draw context from the overall structure of the text in addition to the individual words. This is possible because of the recurrent connection between layers (the activation weights are carried forward). I'm using a simple RNN which is shown in Figure 1. My limited knowledge of using deep learning libraries has prevented me from developing more sophisticated networks but as I will showcase in the results later, even this simple RNN delivers decent results which showcase its potential to be a better classifier than traditional methods, provided I can develop an architecture sophisticated enough. The maximum input length I've considered is 250 and the maximum number of words for the embedding layer is set to 500. From my testing, I've found that this produces the best results. The bidirectional layer has 128 LSTM units in it. These enable the recurrent connections by looking at both the text that came before the current token and the text that follows it. The LSTM method allows it to "remember" relationships between tokens that may be far apart in text. I'm using a relu activation for the first dense layer and a sigmoid activation for the final output layer.

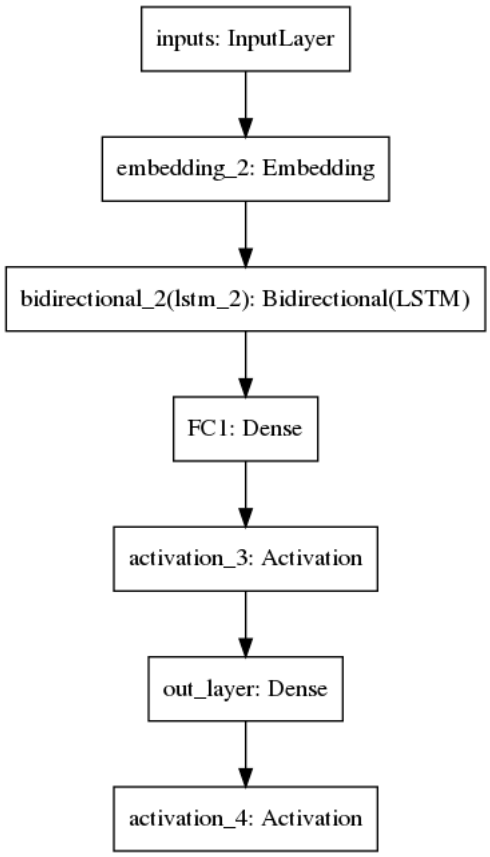

Figure 1: A Simple RNN for classification

4 Results

As mentioned above in the Methodology section, I've conducted tests using multiple approaches. The approaches are broadly split into 2 parts - All feature combinations (Character, Word and POS n-grams) and Only word features, and Doc2Vec vectors and training using RNN. The following sub-sections describe the results for each of them.

4.1 All features and Word features

Here I conducted tests using combinations of 2 - 8 grams boolean and TF n-grams along with χ^2 feature selection the top 50000 features. The results are presented in Table 3.

Features	Value Assignment	MLP	XGBoost
All features	Boolean	68.37	67.32
All features	TF	71.22	66.86
Word features	Boolean	65.41	61.7
Word features	TF	70.57	65.95

Table 3: 10-fold CV accuracies for different features settings and classifiers

We can see from the table that All features with the term frequency value assignment yield the best results. Specifically, the best results came from using 2, 3 and 4 grams. The test set accuracies are mentioned in Table 4. We can observe that just using the word counts got an accuracy of 70.57% which is very close to the best score I got of 71.22% from using all features with feature selection. This just goes to show that the words are the most important features of this data set and therefore deep learning methods have massive potential in improving these accuracy scores because they directly work on these words

and automatically extract features from the text.

Features	Value Assignment	MLP	XGBoost
All features	Boolean	70.22	66.01
All features	TF	71.84	61.48
Word features	Boolean	65.37	60.51
Word features	TF	70.87	67.31

Table 4: Test set accuracies for different features settings and classifiers

4.2 Doc2Vec and RNN

The Doc2Vec results are better than those I got using XGBoost and given that I had to do almost no feature engineering myself, this showcases how effective such methods can be. However, in this context, Doc2Vec didn't prove to be too useful. The results from Doc2Vec are shown in Table 5 and Table 6.

Parameters	MLP	SVM
DBOW, Vector size 600	68.48	66.68
DBOW, Vector size 300	68.41	67.97
PV-DM, Vector size 600	67.03	67.65
PV-DM, Vector size 300	68.01	67.79

Table 5: 10-fold CV accuracies for different settings and classifiers

Parameters	MLP	SVM
DBOW, Vector size 600	68.28	69.57
DBOW, Vector size 300	67.63	67.31
PV-DM, Vector size 600	66.34	65.69
PV-DM, Vector size 300	67.96	66.99

Table 6: Test set accuracies for different settings and classifiers

For the simple RNN, the results were even worse but given the basic nature of the network that was to be expected. When I built the network initially, I had added a Dropout layer to the network to prevent overfitting given the small size of the data, but removing that layer actually improved both the CV and test accuracies indicating that the network was in-fact underfitting the data. I tried adding additional units and layers but was unable to train the network due to resource constraints on my personal computer. Running the blogs through the RNN with the settings in Section 3.4 resulted in a 59.78% and 58.89% 10-fold CV and test set accuracy respectively. But just the fact that a network this simple was so much better than random guessing indicates its potential.

5 Challenges faced

As I mentioned earlier, testing the model was a challenge. If one is not careful subtle bugs can creep in which can result in impressive results which are unfortunately false. I discuss one such issue that almost got me in the next section. Additionally, training the models and extracting features was a slow process given the limited hardware capacity I've got and the amount of GPU compute power I have. This did hamper my progress because I had to wait a good amount of time for the validation results of my model and then tune my model again to work on it more.

6 Learnings

A key take-away for me from the project was the importance of rigorous cross-validation and testing methodologies for machine learning models. While working on the project, I was seeing extremely accurate predictions (90% in 10-fold CV) being made by my models. I later realized that there was a subtle issue with my code in that I had performed feature selection on my entire dataset and was then splitting it into train/val/test sets. After speaking with the professor I realized that this is a mistake because this approach has resulted in the model losing its ability to generalize well to unseen data. In other words, my model was overfitting by a huge margin and would've probably performed terribly on any new unseen data (outside of the current corpus) even though it performed extremely well on the current data.

Additionally, I learned how to actually apply machine learning principles to a real world problems and how I can build experiments and approaches to better solve the problem at hand.

7 Future Work

I would like to take a deeper dive into tackling this problem of gender classification using more sophisticated deep learning approaches. (Bartle and Zheng, 2015) have implemented a promising approach using windowed recurrent-convolutional neural networks whose performance is comparable to the state-of-the-art. I would like

to try and build on their approach to improve the performance on this task even further.

8 Conclusion

Working on this class project was certainly an exciting endeavor and I learned a lot about how to apply machine learning principles to solve real world challenges. Although I was unable to beat the baselines set in (Mukherjee and Liu, 2010), I realized the amount of work and effort it takes to make a good machine learning model to solve a problem and it's not simply throwing our data into an algorithm as beginners like me would believe. I hope to keep working on improving my deep learning skills and applying them to solve more real world problems.

Acknowledgments

I would like to thank Dr. Arjun Mukherjee for his support and guidance towards the completion of this project.

References

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

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

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

Shlomo Argamon, Moshe Koppel, James W Pennebaker, and Jonathan Schler. 2007b. Mining the blogosphere: Age, gender and the varieties of self-expression. *First Monday*, 12(9).

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

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

François Chollet et al. 2015. Keras. https://keras.io.

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.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *EMNLP*.

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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of age and gender on blogging. pages 199–205.