Gender Classification with Deep Learning

Aric Bartle, Jim Zheng

Abstract

For our project, we consider the task of classifying the gender of an author of a blog, novel, tweet, post or comment. Previous attempts have considered traditional NLP models such as bag of words and n-grams to capture gender differences in authorship, and apply it to a specific media (e.g. formal writing, books, tweets, or blogs). Our project takes a novel approach by applying deep learning models developed by Lai et al to directly learn the gender of blog authors. We further refine their models and present a new deep learning model, the Windowed Recurrent Convolutional Neural Network (WRCNN), for gender classification. Our approaches are tested and trained on several datasets: a blog dataset used by Mukherjee et al, and two datasets representing 19th and 20th century authors, respectively. We report an accuracy of 86% on the blog dataset with our WRCNN model, comparable with state-of-the-art implementations.

1 Introduction

There has been rising interest in the problem of gender classification of text, especially in the social media and marketing domains. Much of this is due to the growing sources of user information, ranging from short tweets and comments to longer blog post and online novels. Existing systems mainly use features such as words, word classes, and POS (part-of-speech) n-grams for classification learning. However, none use deep learning, nor extend their models to a variety of sources, for example from the blogging to the media sphere.

Our effort represents two contributions. First, we apply deep learning to datasets spanning several media (e.g. blogs and literature from several centuries). In this approach, our model directly learns to predict gender based on the surrounding context of words. This technique was proposed by Lai et al[5] for a wide variety of text classifications, and we extend their previous work to the topic of gender classification.

Second, we develop a model based on the RCNN developed by [5]. for gender classification. Our model obtains an accuracy score comparable to the state-of-the-art models without much fine-tuning.

2 Related Work

There have been several papers over the years studying the topic of gender classification in text [1,2,3]. These papers treat the problem as a classical machine learning one, and train a linear or non-linear classifier using handcrafted, word-based features on a variety of textual sources. [11] studied gender and text relationships in formal writing using the British National Corpus [11], and discovered that there was a clear difference in writing styles of male and female authors. [1] looked at the effectiveness of tweets to identify author gender. Their approach used n-grams concatenated with author's profile information to predict an author's gender, obtaining around 77% accuracy using texts alone, and notably above 90% accuracy with all features included. The current state-of-the-art using pure textual features was demonstrated by Mukherjee and Liu [2], who looked at content words, dictionary-based content analysis results, and POS (part-of-speech) tags for blog posts. Their hand-crafted features consisted of the following categories:

- **Frequency Measure:** Frequency of various parts of speeches. The F Measure is based upon the observation that males and females tend to have different preferences in frequency of types of parts of speech.
- **Stylistic Features:** These features are characterized by the words used particularly in the blog context.
- **Gender Preferential Features:** These features represent the tendency that females us more emotionally intensive adverbs and adjectives where as males tend to be more punctuated.
- **POS pattern features:** These features represent patterns of POS tags.

They achieved a start-of-the-art performance of 88% using these features and an SVM for classification.

At the same time, there have been attempts at applying deep learning to topic classification, opinion mining. Mikolov, Yih, and Zweig [9] showed that vector representations learned by recurrent neural networks can accurately represent syntactic and semantic regularities within text. Mikolov et al. [10] notably demonstrated word2vec as a simple way of learning these representations. Incidentally, the much simpler paragraph2vec model introduced in [6] produced better results on a variety of text classification tasks but at the cost of additional processing time.

Authors have also considered convolution networks for text classification. [8] used a convolution neural network for sentiment classification of sentences. The work was extended in [7] to classification of documents. The recent work of Lai, Xu, Liu, and Zhao [5] applied recurrent convolutional neural networks to topic classification, and obtained high accuracy on a variety of text classification tasks. But of these approaches, none have specifically addressed the problem of gender classification using deep learning.

3 Technical Approach and Models

Our method can be viewed primarily as an extension of the RCNN model [5]. We will first summarize that model in more detail before developing our extension.

3.1 Recurrent Convolution Neural Network

At a high level the RCNN seeks to represent each word as a context, which is a concatenation of the word's own embedding and the contexts of the words left and right of it. Each context is then transformed through a non-linear hidden layer and elementwise max pooled with all other word contexts to result in a single (document) vector that describes the example. This document vector is then fed through a standard softmax output layer for prediction. We now describe the steps in detail.

Each word w_i is associated with a vector embedding, $\mathbf{e}(w_i)$, in the embedding matrix $\mathbf{E} \in \mathbb{R}^{e \times V}$ where V is our vocabulary size and e is the word embedding dimension. The left and right contexts of a word are defined by a bidirectional recurrent neural net as

$$\mathbf{c}_l(w_i) = f(\mathbf{W}^{(l)}\mathbf{c}_l(w_{i-1}) + \mathbf{W}^{(sl)}\mathbf{e}(w_{i-1}))$$
$$\mathbf{c}_r(w_i) = f(\mathbf{W}^{(r)}\mathbf{c}_r(w_{i+1}) + \mathbf{W}^{(sr)}\mathbf{e}(w_{i+1}))$$

where $\mathbf{W}^{(l)}, \mathbf{W}^{(r)} \in \mathbb{R}^{c \times c}, \mathbf{W}^{(sl)}, \mathbf{W}^{(sr)} \in \mathbb{R}^{c \times e}$, and f is an activation function. The matrices $\mathbf{W}^{(l)}, \mathbf{W}^{(r)}, \mathbf{W}^{(sl)}, \mathbf{W}^{(sr)}$ serve to transform each word's context into a hidden layer of dimension c. The context for a word w_i is then defined as

$$\mathbf{x}(w_i) = [\mathbf{c}_l(w_i) \;\; \mathbf{e}(w_i) \;\; \mathbf{c}_r(w_i)]^T$$

The intuition for this definition is that by using contextual information for all words left of w_i and all words right of w_i, we may better capture the meaning of the word through even long range dependencies compared to models that use a fixed window size. We also note that $\mathbf{c}_l, \mathbf{c}_r$ can be computed in linear time with simple forward and backward passes.

After this stage, each word is fed through a standard affine transform and a tanh activation to yield a latent semantic vector:

$$\mathbf{y}^{(2)}(w_i) = \tanh(\mathbf{W}^{(2)}\mathbf{x}(w_i) + \mathbf{b}^{(2)})$$

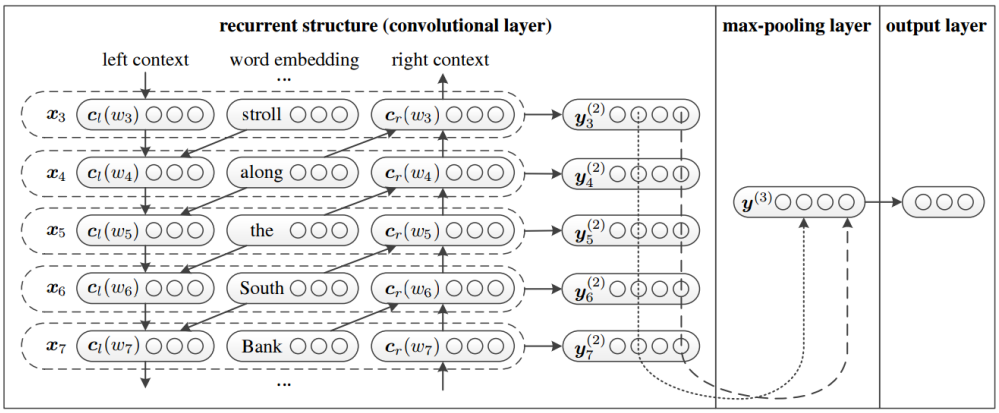

Figure 1: Recurrent Convolutional Neural Network

where $\mathbf{W}^{(2)} \in \mathbb{R}^{H\times(2c+e)}, b^{(2)} \in \mathbb{R}^H$. Finally, all such latent semantic vectors may be pooled together to yield a document vector:

$$\mathbf{y}^{(3)} = \max_i \mathbf{y}^{(2)}(w_i)$$

where the max here is understood to be elementwise. We can view the network so far constructed as a convolutional neural net where the recurrence defines the convolutional layer and we apply a max pooling layer to generate a vector representation. We finally may define the output layer using the standard softmax:

$$\mathbf{y}^{(4)} = \text{softmax}(\mathbf{W}^{(4)}\mathbf{y}^{(3)} + \mathbf{b}^{(4)})$$

and $W^{(4)} \in \mathbb{R}^{2\times H}, \mathbf{b}^{(4)} \in \mathbb{R}^H$. Specifically, we are looking for binary classification so we have the output layer of size 2. Figure 1 depicts the model in detail.

We then minimize the cross entropy loss defined for a set of documents, $D = \{d_1, d_2, ...\}$, labeled as either male or female. We use L^2 regularization on the weight matrices $\mathbf{W}$ (with a value of .001), and train using stochastic gradient descent. As in [5] was used pre-trained embedded word vectors. For all of our examples we found the hyperparameters $H = 100, c = 50$ to be optimal.

3.2 Windowed Recurrent Convolution Neural Network

An underlying problem with the previous network is that it treats the entire document as a training example. When blog posts or even paragraphs are hundreds of words, it becomes less clear that such a global approach as a bidirectional RNN will work effectively. We also note that often times individual sentences will be highly predictive of a person's gender at least our data set conforms to this. With these considerations, we consider a modified model of the RCNN.

Let $s_{j,k}$ represent the jth sentence in document k and $w_{i,j,k}$ the ith word of sentence j in document k, then

$$\begin{aligned}
\mathbf{c}_l(w_{i,j,k}) &= f(\mathbf{W}^{(l)}\mathbf{c}_l(w_{i-1,j,k}) + \mathbf{W}^{(sl)}\mathbf{e}(w_{i-1,j,k})) \\
\mathbf{c}_r(w_{i,j,k}) &= f(\mathbf{W}^{(r)}\mathbf{c}_r(w_{i+1,,j,k}) + \mathbf{W}^{(sr)}\mathbf{e}(w_{i+1,j,k})) \\
\mathbf{x}(w_{i,j,k}) &= [\mathbf{c}_l(w_{i,j,k}) \;\; \mathbf{e}(w_{i,j,k}) \;\; \mathbf{c}_r(w_{i,j,k})]^T \\
\mathbf{y}^{(2)}(w_{i,j,k}) &= \tanh(\mathbf{W}^{(2)}\mathbf{x}(w_{i,j,k}) + \mathbf{b}^{(2)}) \\
\mathbf{y}^{(2)}(s_{j,k}) &= \max_i \mathbf{y}^{(2)}(w_{i,j,k}) \\
\mathbf{y}^{(3)}(s_{j,k}) &= \tanh(\mathbf{W}^{(3)}\mathbf{y}^{(2)}(s_{j,k}) + \mathbf{b}^{(3)}) \\
\mathbf{y}^{(4)}(d_k) &= \max_j \mathbf{y}^{(3)}(s_{j,k}) \\
\mathbf{y}^{(5)}(d_k) &= \text{softmax}(\mathbf{W}^{(5)}\mathbf{y}^{(4)}(d_k) + \mathbf{W}^{(4)})
\end{aligned}$$

The model above identifies the strongest activation in each sentence/window and then the entire document. We note, however, that the model also disregards the sentence order; nonetheless, we believe that individual sentence responses are more valuable than sentence interactions in gender classification.

As with before we minimize the cross entropy loss, use L^2 regularization on the weight matrices $\mathbf{W}$, and train using stochastic gradient descent. We obtained best results with $c = 50$ and both of the hidden layers with 100 dimensions.

4 Dataset and Preprocessing

Our dataset is split into two categories: blogs, and 19th/20th century books. We treat each of the three data sources separately, so as to group authors by century. We believe that this leads to largely consistent styles within these groups, since there can be significant vocabulary and grammatical differences between, say, Jane Austen and Langston Hughes. We select from an English corpus only, and discard parts of blogs and books in any other language during our data gathering process.

4.1 Blogs

For blogs, we use the same dataset as [2] in order to draw comparable conclusions. This data set represents text scraped from the RSS feeds of Google Blogspot and consist of: 1679 male blog posts and 1548 female blog posts. This dataset includes several interesting features not found in books, namely emoticons, irregular punctuation / grammatical errors.
Sample blog post from this dataset:

"Oh friends, it's finally here! I thought the month between Christmas break and midwinter break wouldn't be too slow and awful. I told myself I wouldn't take any days off, and I succeeded in that. There was a pretty rough stretch for a bit, though, with that bunch of teachers leaving. Loud Leo left our class a few weeks ago. A new boy came to the class, but has apparently left again. Buster, whom I have not yet mentioned, has remained a serious detriment to the class's focus and my..."

4.2 Books

For books, we pull from Project Gutenberg's book excerpts, freely available online. We manually downloaded and classified authors by century and gender, and for each, downloaded a length (500 word+) excerpt of their selected work.

Bias While gathering this dataset, we noticed several biases, mainly pertaining to the low number of female authors represented for the 20th century. This is a limitation of the API, and we carefully select our works and examples to obtain similar numbers of male and female examples

4.3 Extraction

After gathering the raw data, we wrote a custom parser to perform text extraction and sanitization. Notably, the blog dataset contained emoticons and additional characters in an unrecoverable unicode format, and we had to remove such characters.

For each labeled example, we extract successive windows of K sentences. We let K be 12 (average paragraph size) for our main experiments. We do such an extraction for two reasons. First, we can expand our training set by doing so, particularly for our novels. Second, we can enforce better length consistency across examples. We want to avoid any biases that may be produced as a result of length.

In all cases, we break our extracted data up into training, dev, and test sets. We choose a split of 80% training, 10% dev, and 10% test. Our final data set has the specifications:

	Male	Female
Blogs	2198	2043
19th	1192	1087
20th	1167	1201

Table 1: Dataset sizes

5 Experiments

5.1 Method Comparison and setup

In order to compare our approach, we considered four other models besides the RCNN for performance comparisons:

Bag of Words: The BOW model has been successful in many document related tasks. We use the BOW model to generate for each document a feature vector and classify with an SVM.

Average Embedding: We consider a simple single hidden layer network

$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(2)} \tanh(\mathbf{W}^{(1)}\mathbf{h} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})$$

where $\mathbf{h}$ is the average of the word vectors in the documen. This is similar to [4] who used the average embedding for global context scores. We used hidden dimensions of 50 and L^2 regularization with value $.001$.

paragraph2vec: The paragraph2vec model [6] had achieved state-of-the-art performance on document classification tasks. Using the model, we extract a paragraph/document vector for each document and use an SVM to classify the resultant document vectors.

POS features: The work of [2] achieves state-of-the-art performance on the blog dataset. We compare our approach by directly running on the blog dataset.

5.2 Results and Discussion

	BOW	Average Embeddings	paragraph2vec	RCNN	WRCNN	POS Features
blogs	60%	74%	72%	81%	86%	88%
19th	52%	61%	62%	67%	73%	-
20th	53%	62%	63%	69%	71%	-

Table 2: Accuracy for experiments

We list the various results in the table above.

- All performance results are significantly lower for the literature datasets. This is a result of the fact that the blog dataset is inherently biased. The bias stems from the fact that the male based examples tend to deal with situations associated with more masculine settings, and similarly, female based examples represent more feminine situations. On the other hand, the literature dataset tends to cover more similar subjects and has less environmental bias.
  When looking at individual excerpts from the literature dataset, it is clear that the differences between male and female authors are far more subtle and would require far more contextual understanding. It would be informative to see how well [2] does on these datasets.
- BOW classification gives poor results on the blog dataset and mostly random results on the literature datasets. Although BOW has been highly successful in such cases as spam classification, those cases generally involve a distinct of key words. In both blogs and especially literature, there is not really a set of key words.
- The average embedding model and paragrpah2vec do reasonably well given their more simplistic nature. As expected they easily out perform BOW as they can capture more contextual information and may have less problems with data sparsity.

- The RCNN model allows for better learning of word order and sentence/paragraph structure compared to models like paragraph2vec that can't directly take into account sentence and paragraph structures.
- Our extension of the RCNN model, WRCNN, does on average 4% better. This result leads credibility to our assertion that individual sentences may output a strong signal and be beneficial for classification. To further our argument, we created several instances of the data at different K values (number of sentences per example). We found that increasing K resulted in more accurate predictions of WRCNN compared to RCNN.
- The results produced by our new model are notably the best except for the POS features. Nonetheless, we still achieve comparable performance and believe that better performance could be achieved if the unicode characters had been recoverable. Additionally, techniques such as dropout could have been used to potentially inhance performance.

5.3 Implementation

We implemented our methods using a variety of different tools. For BOW we created a simple python implementation and used libsvm for classification. In paragraph2vec we used an existing implementation of word2vec that had support for computing paragraph vectors. We used this code to extract paragraph/document vectors and then used libsvm for classification. Finally, we used Theano for the implementation of the Average Embedding model and both the RCNN and WRCNN.

As an aside, we tested our implementation of the RCNN on the Stanford Tree Bank which [5] also used. We obtained the accuracy of 46.91%, which is comparable to what was achieved in [5].

6 Conclusion and Future Work

In this paper we considered the problem of author gender classification across several media. We extended the RCNN model developed by Lai et al [5] for topic classification of documents to obtain a baseline accuracy. We improve upon this model by using max pooling on sentences and obtained a classification accuracy comparable to the state-of-the-art results achieved by Mukherjee and Liu [2], and we establish a baseline for this dataset of novels. Future work can include additional enhancements to our neural network, such as k-max pooling over sentences [8] as well as further investigation across several centuries of literature, other types of media, and individual authors.

References

[1] John D. Burger, John Henderson, George Kim, Guido Zarrella. Discriminating Gender on Twitter. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011

[2] Arjun Mukherjee, Bing Liu. Improving Gender Classification of Blog Authors. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.

[3] Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. Gender Attribution: Tracing Stylometric Evidence Beyond Topic and Genre. *Conference on Computational Natural Language Learning*, 2011

[4] Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, 873–882.

[5] 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

[6] Quoc Le, Tomas Mikolov. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning, Beijing, China*, 2014

[7] Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, Nando de Freitas. Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network.

[8] Nal Kalchbrenner, Edward Grefenstette, Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. arXiv:1404.2188.

[9] Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. Microsoft Research, Redmond, WA 98052

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases. NIPS (2013)

[11] Shlomo Argamona, Moshe Koppel, Jonathan Finec, Anat Rachel Shimoni. Gender, Genre, and Writing Style in Formal Written Texts, Text-Interdisciplinary Journal, 2003