Authorship Attribution for Neural Text Generation

Adaku Uchendu **Thai Le** **Kai Shu**[†] **Dongwon Lee**

The Pennsylvania State University, University Park, PA, USA
{azu5030, thaile, dongwon}@psu.edu

Illinois Institute of Technology, Chicago, IL, USA[†]
kshu@iit.edu[†]

Abstract

In recent years, the task of generating realistic short and long texts have made tremendous advancements. In particular, several recently proposed neural network-based language models have demonstrated their astonishing capabilities to generate texts that are challenging to distinguish from human-written texts with the naked eye. Despite many benefits and utilities of such neural methods, in some applications, being able to tell the "author" of a text in question becomes critically important. In this work, in the context of this *Turing Test*, we investigate the so-called *authorship attribution* problem in three versions: (1) given two texts T_1 and T_2, are both generated by the same method or not? (2) is the given text T written by a human or machine? (3) given a text T and k candidate neural methods, can we single out the method (among k alternatives) that generated T? Against one human-written and eight machine-generated texts (i.e., CTRL, GPT, GPT2, GROVER, XLM, XLNET, PPLM, FAIR), we empirically experiment with the performance of various models in three problems. By and large, we find that most generators still generate texts significantly different from human-written ones, thereby making three problems easier to solve. However, the qualities of texts generated by GPT2, GROVER, and FAIR are better, often confusing machine classifiers in solving three problems. All codes and datasets of our experiments are available at: https://bit.ly/302zWdz

1 Introduction

Recent rapid advancements in deep learning technologies have enabled the generation of realistic artifacts (e.g., Deepfakes) that are difficult to distinguish from genuine human-generated artifacts. In the *text* domain, which is the main focus of this work, similarly, the advancement of *Natural Language Generation* (NLG), especially those based on neural language models, has led to the inundation of realistic text generation.

As novel NLG techniques become more sophisticated and prevalent, corresponding pitfalls and risks of such technologies also increase. Adversaries may use such technologies to generate realistic artifacts to trick naive users in fraudulent activities (e.g., machine-generated chatbot conversation in a phishing scam or deepfake-based disinformation campaign). Therefore, the need to distinguish machine-generated texts from human-written ones, so-called the *Turing Test*, naturally arises. Furthermore, in some security applications, merely being able to identify machine-generated text may not be sufficient. Instead, a more critical solution would be to tell *which* NLG method among many candidates has generated a given text in question–so-called the *Authorship Attribution (AA)* problem. To improve our understanding of this newly-emerging problem, we empirically investigate three versions of the AA problem in this paper. For all three versions, we assume that there are k different NLG methods[1].

Problem 1 (Same Method or Not) *Given two texts T_1 and T_2, determine if both T_1 and T_2 are generated by the same NLG method (or human) or not.*

Problem 2 (Human vs. Machine) *Given a text T_1, determine if T_1 is written by human or generated by any k NLG methods.*

Problem 3 (Authorship Attribution) *Given a text T_1, single out one NLG method (among k alternatives) that generated T_1.*

We model P1 and P2 as the binary classification problem, while P3 as the multi-class classification

[1] In the future, we envision that k can be huge, say 1000, but for this experiment, we set $k = 8$.

problem. All three problems are related, with several motivations as follows.

First, solutions to P1 may be useful when one needs to determine the plagiarism or identity theft issue of an NLG method. For instance, suppose GPT2 becomes very powerful in the near future so that other NLG methods may even try to mimic the characteristic features in the GPT2-generated texts. Then, a solution to P1 can determine if two texts in question are both generated by GPT2 or not. Second, as NLG methods become ubiquitous, the threat of generating misinformation at scale increases naturally. Thus, using solutions to P2, being able to accurately distinguish between machine- and human-generated texts is required to mitigate such security risks that NLG methods could pose. Finally, as to P3, as the number of state-of-the-art NLG methods increases, it will be beneficial not only just to separate them into two camps but also find out which generators are used. Furthermore, knowing each generator's writing signature or style moves us closer to quenching the security threats that they may introduce.

2 Related Work

2.1 Features for Authorship Attribution

Predicting an author based on their writing signature is called Authorship Attribution (AA). This AA problem has been previously and even recently solved with n-grams (Sharma et al., 2018; Sari et al., 2017; Shrestha et al., 2017; Proisl et al., 2018; Kestemont, 2014; Zečević, 2011; Li et al., 2014). Next, as complex datasets emerge, other techniques such as POS-tags (Ferracane et al., 2017; Sundararajan and Woodard, 2018; Hitschler et al., 2017), topic modeling (i.e. LDA, AT and DADT) (Seroussi et al., 2014, 2012, 2011), POS-Noise (Halvani et al., 2020) and LIWC (Uchendu et al., 2019; Li et al., 2014) are explored and used to solve the AA problem. However, Ferracane et al. claim that n-grams and POS-tags are not sufficient for solving the AA problem, and sometimes negatively impact classifiers' performance. Therefore, they recommend using discourse embedding features.

Furthermore, Zheng et al. attempt to solve the AA problem with online messages by investigating four types of writing-style features (i.e., lexical, syntactic, structural, and content-specific features). Structural and content-specific features were the best to assign authorship (Zheng et al., 2006).

Next, Kestemont examines the use of content and function words as relevant features for AA. Van Cranenburgh focuses on content words and so parse phrase-structures. Consequently, Hoenen and Schenk claim that word pairs could make strong features and extract function words, content words, similarity and relatedness. They find function words to be the most robust feature (Hoenen and Schenk, 2018). To establish distinct writing styles further, Solorio et al. extract lexical, syntactic, and stylistic features using bag-of-words (freq. of unigrams), POS-tags, Dependency relations, and Chunks (unigram freq.), respectively for the AA problem. With the new wave of nuanced techniques to solve the AA problem, Tschuggnall and Specht use syntax tree for each sentence to analyze grammar. Finally, Shao et al. use readability scores to distinguish human-written texts from machine texts, another form of AA. The results suggest that readability is a vital feature for distinguishing authors.

2.2 Classifiers for Authorship Attribution

Several well-established classical machine learning classifiers have been applied to the AA problem, including Naive Bayes (Howedi and Mohd, 2014; Baron, 2014), SVM (Solorio et al., 2011; Hou and Huang, 2017; Shao et al., 2019), Conditional Tree (Sharma et al., 2018), Random Forest (Hou and Huang, 2017; Alshaher and Xu, 2020; Sharma et al., 2018), and KNN (Alshaher and Xu, 2020). However, due to improvements in neural networks, recently, CNN (Convolution Neural Network) is said to be even more suited for the AA problem (Ferracane et al., 2017; Hitschler et al., 2017; Boumber et al., 2018). CNN architecture is better suited to represent the characteristics of each author. Consequently, Ferracane et al. improve the CNN's use with discourse features (i.e., n-grams and POS-tags). Ren and Ji further improve upon CNNs with the use of word embeddings to represent texts. Lastly, RNNs have also been shown to be well-suited for representing the authors' distinct writing styles (Alsulami et al., 2017).

2.3 Applications of Authorship Attribution

The applications of AA are vast and include: assigning authorship to literature/text, and ascertaining the demography of an author (e.g., age, gender, native language) (López-Monroy et al., 2020). AA can also be applied to predicting author(s) of source code (Simko et al., 2018), chatbot detection (Uchendu et al., 2019), and even detecting authors intentionally trying to mask their writing style (Juola, 2012; Sánchez-Junquera et al., 2020). Finally, our work bears similarity to (Manjavacas et al., 2017), which investigates the stylistic properties of different neural text generation techniques (i.e., Ngram-based and RNN-based).

3 Generation of Texts

We have nine text generators–i.e., one human writer and eight neural machine generators. All eight neural generators require a short prompt to begin their generation and the number of words to generate. These eight generators were chosen because we found that they had the best pre-trained models for our task. We used the titles of news articles (written by human journalists) as the prompt and set 500 as the number of words.

1. **Human**. We collected recently-published news titles and contents in mostly Politics–819 from CNN, 132 from Washington Post, and 113 from the New York Times. As professional reporters write these news articles, they represent human-written texts. Then, we used the news titles as the prompts for other neural methods.

2. **CTRL**. Also known as "Conditional Transformer Language Model For Controllable Generation," CTRL[2] is a huge language model with 1.63 billion parameters (Keskar et al., 2019). The model was trained on control codes to guide the styles and contents of generated texts. Among the 50 control codes available, we used the *News* control code to generate long articles.

3. **GPT**. The OpenAI GPT is built with Transformers. It was trained and modeled after a simple concept - to predict the next token, given the previous token (Radford et al., 2018). We used the medium GPT model with 345 million parameters since it was computationally less expensive while still being able to generate comparable results. We used the Transformer text generation setup by huggingface[3].

4. **GPT2**. We also used the GPT2 model with 774 million parameters. We used the gpt2 wrapper[4] to generate texts.

5. **GROVER**. Grover is another large language model, explicitly trained to generate political news (Zellers et al., 2019). It uses the same template as news outlets such as CNN and the New York Times. Grover uses the same architecture as GPT2 and the same concept of predicting the next token, given previous tokens. We used code from repo[5] to generate texts.

6. **XLM**. The Cross-lingual Language Model (XLM) is another generative language model (Lample and Conneau, 2019). Unlike other language models, XLM is trained for the task of cross-lingual classification. We generated texts from the English language model, using the same setup in huggingface as GPT.

7. **XLNET**. XLNET (Yang et al., 2019) improves language modeling by introducing bidirectional contexts. This technique involves a generalized auto-regressive pre-training method and adopts the Transformer-XL framework into pre-training. XLNET achieved state-of-the-art results, outperforming BERT on 20 tasks. We also used the huggingface Github repo to generate texts.

8. **PPLM**. The "Plug and Play Language Models (PPLM)" is another language model that improves upon GPT2 by fusing the medium model with bag of words models (Dathathri et al., 2019). We used the *Politics* bag of words model to generate texts', using the code[6], and used the perturbed version.

9. **FAIR**. Facebook's FAIR has three language models of English, Russian, and German (Ng et al., 2019). For our task, we used the English language model built with FAIRSEQ sequence modeling toolkit[7].

Table 1 shows the snippets of nine texts for the identical prompt message–i.e., one written by news reporters and eight by different neural language models. Table 2 shows the summary statistics of nine generated texts. Note that CTRL tends to generate the longest texts (in terms of the number of

[2] https://github.com/salesforce/ctrl

[3] https://github.com/huggingface/transformers

[4] https://github.com/minimaxir/gpt-2-simple

[5] https://github.com/rowanz/grover

[6] https://github.com/uber-research/PPLM

[7] http://shorturl.at/swDHJ

PROMPT	Putin and Xi are using the coronavirus crisis to extend their control. Across the world, Trump is struggling to keep up
Human	. . . Donald Trump's shock announcement shutting down travel from much of Europe came with the customary back-pedaling common to many of his hastily announced initiatives. At first, Trump said trade and travel would be targeted, in his latest attempt to stem the spread . . .
CTRL	. . . . President Vladimir Putin, meanwhile, says there was no reason why Moscow would spread such diseases. But both leaders have been unable so far to . . . [read more] what do you think about this story? do we need another Ebola scare or two? . . .
GPT	. . . with them in a war that has been going on for decades now - but he's still not strong enough. " " i'm sorry about your family. it must be hard being so far away from home. " she said softly. " yes, we're all very lonely here. there wasn't much time before . . .
GPT2	. . . Putin's paranoia over the contagion of Ebola is a common theme. As his recent visit to Washington shows, he's playing a dangerous game. And here's what Putin wants Putin wants to use Ebola as a pretext for imposing further Russian sanctions on the West. . . .
GROVER	. . . hours after world cup workers hand-picked by Putin to fix problems faced by workers in turkey and Venezuela, a new threat to trump is receding.According to Wikileaks, the Coronavirus outbreak is receding and the Russian leader, Putin,. . .
XLM	. . . with Russia alone ruling over Europe while china under Mao Zedong rules Asia in turn which makes it difficult for him politically but financially as well being president / CEO of America first bank & chairman's nominee at CIA . . .
XLNET	. . . with china or Russia in his efforts at de-destructuring America from the financial crisis that has taken place over the past two years of this century as well a global economic downturn that will continue for many more months into next year . . .
PPLM	. . . the latest developments in Syria, Ukraine, Russia's intervention in the middle east and china's rise to global power. In Moscow, a group of Russian diplomats, who did not want their real names to be reported on the record. . .
FAIR	.. He has ordered his federal agencies to step up efforts to combat it and to warn of new threats. in Washington, Trump's national security adviser, h.r. mcmaster, said on Monday the U.S. government had seen signs of the bug and that countries in the Middle East were on high alert. . . .

Table 1: Snippets of nine texts using the titles of human-written articles as the prompt to neural methods.

words), while GPT2 tends to generate the shortest texts. Both XML and XLNET generated the texts with very long sentences.

4 Linguistic Analysis

We first conduct a psycholinguistics study to analyze different linguistic features of generated texts. The result is summarized in Table 3.

First, we use *Flesch Reading Ease* and *Flesh-Kincaid Grade* to gauge generated texts' readability. *Flesch Reading Ease* generates a score between 0 and 100, such that post-college level yields a score between 0-30, college-level yields 31-50, high-school level yields 51-70, middle school yields 71-90, and 5-th grade level of reading and below yields 91-100. These seven reading levels also go from a scale of very-difficult-to-understand due to the level of sophistication to very-easy because it is the grade level of readability. Therefore, obtaining a post-college level (i.e., low score) is uncommon and impressive if a machine generates such texts. On the other hand, the *Flesh-Kincaid Grade* generates a score representing the U.S. grade level of education (the higher, the more sophisticating). For instance, text given a 10.8 score suggests that its author can be in the 11-th grade and about 16-17 years old.

Next, we use *Linguistic Inquiry and Word Count (LIWC)* (Pennebaker et al., 2001) to capture the psycholinguistics features. LIWC has 93 features, of which 69 are categorized into: Standard Linguistic Dimensions (e.g., pronouns, past tense), Psychological Processes (e.g., social processes), Personal concerns (e.g., money, achievement), and Spoken Categories (e.g., assent, nonfluencies) (Uchendu et al., 2019). Table 3 includes top-3 distinguished LIWC features among all generation methods. A high *LIWC-Authentic* score means that the author of the text is honest or less evasive. We can observe that GPT and XLNET generates more personal content than GPT2 and FAIR. *LIWC-Analytic* reflects the formality, and logical nature of the text. GPT2,

Measure	Human	Machine							
		CTRL	GPT	GPT2	GROVER	XLM	XLNET	PPLM	FAIR
# of samples	1,066	1,066	1,066	1,066	1,066	1,066	1,066	1,066	1,066
AVG word count	432.31	530.03	345.03	199	356.76	441.32	452.58	228.89	250.42
SD word count	270.82	73.51	10.79	74.15	114.96	34.67	32.59	64.13	39.94
AVG sentence count	26.87	33.02	32.64	15.68	21.64	3.97	5.02	13.53	17.53
SD sentence count	19.49	21.18	5.55	6.99	9.65	1.71	1.97	4.61	4.88

Table 2: Summary statistics of nine generated texts (one by human and eight by neural methods).

Measure	Human	Machine								AVG
		CTRL	GPT	GPT2	GROVER	XLM	XLNET	PPLM	FAIR	
Flesch Reading Ease	37.97	60.97	68.68	54.49	46.63	46.40	48.94	44.97	51.85	51.21
Flesch-Kincaid Grade	12.79	9.58	8.48	10.27	11.53	11.64	11.28	11.66	10.76	10.89
LIWC-Authentic	25.3	54.28	61.66	15.1	23.76	48.06	80.69	34.27	18.77	40.21
LIWC-Analytic	89.81	51.99	40.93	92.59	89.98	78.61	50.46	73.18	92.89	73.38
LIWC-Article	7.98	1.47	3.18	11.87	8.69	0.59	2.03	2.6	10.05	5.38
Entropy	7.81	8.98	8.01	6.52	7.79	8.99	8.91	7.77	7.41	8.02

Table 3: Linguistic features of nine generated texts.

GROVER, and FAIR scores are as high Human, suggesting that they all generate sophisticated texts. *LIWC-Article* shows the usage of *a, an, the*, which are crucial in any formal writing. Similar to *LIWC-Analytic*, GPT2, GROVER, and FAIR score similar to Human. Overall, the patterns among these LIWC features follow our observations that GPT2, GROVER and FAIR generally have higher news generation quality than other machine algorithms. Finally, we also measure the entropy scores of generated texts (Schürmann and Grassberger, 1996). Figure 1 shows the 2-dimensional distribution of generated texts using Principal Component Analysis (PCA) on all psycholinguistic features, with about 70% explained variation. As we can observe a large overlapped portion among generated texts. We expect a non-linear machine learning model (e.g., Random Forest) would perform better than a linear method such as Naive Bayes in classifying the texts according to their generators using these features.

5 Model Architecture

In solving three problems, we compare various relatively-simple neural models' performances, employing different architectures to encode generated texts into representation vectors, which then feed into a fully connected network followed by a *softmax* layer for prediction. Note that our goal is *not* to develop sophisticated neural models to solve three problems. Rather, we want to empirically evaluate how these simple neural models (as baselines) perform in solving three problems.

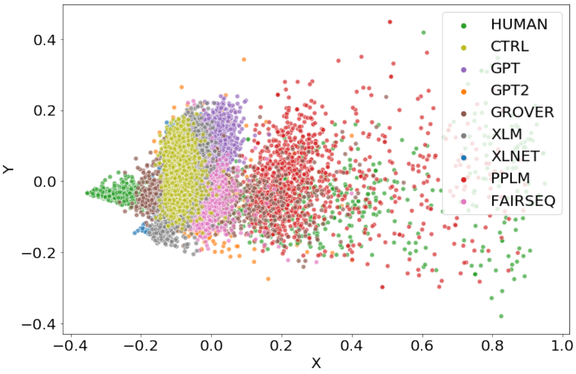

Figure 1: Distribution of generated texts on 2-dimensions using PCA.

1. **Embedding:** This model maps each word in the generated texts to a vector of 300 dimensions, then sums up all resulting vectors as the final representation.

2. **RNN:** This model uses a variant of recurrent neural network (RNN) with a GRU (Cho et al., 2014) layer to model the sequential dependency among words within each of the generated texts.

3. **Stacked_CNN:** This model is inspired by (Zhang et al., 2015), where each of generated texts is encoded by a sequence of six 1D convolutional layers of different kernel sizes. We reduced learning rates from 0.001 to 0.01/0.1.

4. **Parallel_CNN:** Similar to *Stacked_CNN*, but instead of using a stack of convolutional layers, we adopt (Kim, 2014) and use four parallel 1D convolutional layers of different kernel sizes, followed by a max pooling and concatenation operation.

5. **CNN-RNN:** This is a combination of *Stacked_CNN* and *RNN* where each word of a text is first encoded by a stack of two 1D convolutional layers before being input into each step of a GRU layer to model the sequential dependency of the whole text.

Experimenting with these neural models, we split the dataset into the training, validation, and testing parts in 7:1:2 ratio.

6 P1: Same Method or Not

The first version of the problem is to determine whether two given texts are generated by the same method (including human writers) or not. Even if one cannot pinpoint whom the author is for a given text, one may still notice similarities between texts. Therefore, P1 tests the varying capabilities of models to detect such *similarities* between the two texts.

We prepare two datasets of a similar size. In the balanced set, half of text pairs are generated by the same method (e.g., Human-Human or CTRL-CTRL), and the other half are random pairs of the two different methods (e.g., Human-CTRL or GROVER-FAIR). In the imbalanced set, 11% of text pairs are generated by the same method, while the remaining 89% are by different methods (1:8 ratio). Model-wise, we utilize the *Siamese neural network* (Koch et al., 2015) with one of the text encoders in Section 5 to predict whether the two input texts are generated by the same method. Table 4 summarizes the performances. Both RNN and CNN-RNN methods perform the best in the balanced and imbalanced settings, respectively. Recall that the imbalanced setting is more challenging than the balanced as # of positive samples is much smaller. Overall, neural models can identify two texts generated by the same method very well for the balanced setting (F1=0.9813) and reasonably well for the imbalanced setting (F1=0.7869).

7 P2: Human vs. Machine

The second version of the problem determines whether a given text is generated by human or machine (i.e., one of the neural methods). P2 is a type of the *Turing Test*. Despite the recent advancements in neural NLG methods, we hypothesize that there may still be latent differentiating characteristics between human-written and machine-generated texts. Therefore, P2 tests the varying capabilities of different models to detect such *differences* between human and machine writings.

For P2, in addition to five neural models introduced in Section 5, we also tested three known Turing Test models including RoBERTa (Liu et al., 2019) using a similar implementation of *GPT2 Output Detector*[8], GROVER-DETECT (Zellers et al., 2019)[9], and RoBERTa-tuned, which is the RoBERTa that we fine-tuned using 20% of our data. RoBERTa is fine-tuned by adding a classification layer on top of it. Next, the weight of the classification layer is randomly initialized and then trained on the GPT2 output and human written text [10]. Further, we utilize the 20% of the target data we collected to fine-tune the RoBERTa classification model. Note that GROVER-DETECT used in our experiment was trained using only 5K training samples, while its improved version trained with 100K samples is not publicly available. Additionally, *GLTR* is another state-of-the-art Turing tester used to distinguish machine-generated texts from human-generated texts (Gehrmann et al., 2019), although not used in these experiments.

Furthermore, in this setting, we tested both individual case (i.e., one neural method at a time) and collective case (i.e., eight neural methods combined). First, we prepare eight test sets for the individual case, each of which is the balanced test set between human (50%) vs. one neural generator (50%). Table 5 summarizes the performances in those eight individual test sets. For the collective case, on the other hand, we prepare two test sets. In the balanced set, the half of tests are written by human and the other eight neural methods generates the other half. In the imbalanced set, 11% of test texts are written by human, while the remaining 89% are generated by any of the eight neural methods (1:8 ratio). Table 6 summarizes the performances in both balanced and imbalanced settings.

In Table 5, we find that GPT2 generates texts that are almost indistinguishable from human-written

[8] https://github.com/openai/gpt-2-output-dataset/tree/master/detector

[9] https://github.com/rowanz/grover/tree/master/discrimination

[10] https://github.com/openai/gpt-2-output-dataset/

Model	Balanced (1:1)			Imbalanced (1:8)		
	P	R	F1	P	R	F1
Embedding	0.9006	0.8683	0.8841	0.5148	0.7531	0.6116
RNN	**0.9748**	**0.9879**	**0.9813**	0.5439	0.8695	0.6692
Stacked_CNN	0.9509	0.9747	0.9626	0.6269	**0.9269**	0.7479
Parallel_CNN	0.9545	0.9852	0.9696	0.6004	0.8319	0.6974
CNN-RNN	0.9572	0.9750	0.9660	**0.6847**	0.9248	**0.7869**

Table 4: P1: Binary classification performance of "Same Method or Not" on two collective test sets.

Model	CTRL	GPT	GPT2	GROVER	XLM	XLNET	PPLM	FAIR	AVG
Embedding	0.9768	0.9838	0.4044	0.6628	0.6535	0.6551	0.8449	0.5178	0.7124
RNN	**1.0**	0.9930	0.6329	**0.9977**	0.9977	**1.0**	0.9466	0.8812	0.9311
Stacked_CNN	0.9792	0.9815	0.6347	**0.9977**	0.9907	0.9186	0.6457	0.6316	0.8475
Parallel_CNN	**1.0**	**0.9977**	0.6075	0.9536	**1.0**	**1.0**	0.9513	0.9282	0.9298
CNN-RNN	**1.0**	0.9861	0.6626	**0.9977**	0.9699	0.9907	0.7949	0.7018	0.8880
RoBERTa	0.6448	0.6404	0.6407	0.6448	0.6490	0.7185	0.6404	0.6404	0.6524
RoBERTa-tuned	0.9730	0.9881	**0.9792**	0.8894	0.9921	0.9850	**0.9796**	**0.9753**	**0.9702**
GROVER-DETECT	0.7753	0.7319	0.6976	0.8135	0.6929	0.7536	0.7761	0.7616	0.7503
AVG	0.9186	0.9128	**0.6574**	0.8696	0.8682	0.8777	0.8236	0.7547	

Table 5: P2: Binary classification performance in F1 score of "Human vs. Machine" on eight individual test sets. Each column name X indicates an individual balanced test set of HUMAN (50%) and X (50%).

texts (having the lowest average F1=0.6574 across eight models). FAIR is the second (F1=0.7547). Interestingly, we find that RoBERTa-tuned can still differentiate human-written texts from GPT2-generated ones with a high F1 score (0.9792) and has the highest average F1 (0.9702) across all eight datasets. This is likely so because RoBERTa-tuned is fine-tuned on two doses of GPT2 texts (i.e., RoBERTa was already fine-tuned on GPT2 dataset to begin with).

For the performance of collective cases shown in Table 6, RoBERTa-tuned is again the overall winner. It can differentiate human-written vs. machine-generated texts with F1=0.9152 for the balanced setting and F1=0.8489 for the imbalanced setting. Two existing Turing Test models (i.e. GROVER-DETECT and RoBERTa) significantly underperform, although RoBERTa aces in Recall.

8 P3: Authorship Attribution

The third version of the problem is to single out the real author of a given text, among many alternatives (e.g., one human and k neural methods). Therefore, P3 tests different models' varying capabilities to exploit both *similarities* within and *differences* across human and machine writings.

For P3, in addition to five neural models introduced in Section 5, we also tested four classical machine learning models (i.e., Naive Bayes, Decision Tree, SVM, and Random Forest) using psycholinguistic features discussed in Section 4 and four state-of-the-art AA solutions, including POS+CNN-LSTM and POS+LSTM-LSTM (Jafariakinabad et al., 2019), 3-grams + SVM (Sari et al., 2018) and Character n-gram + SVM (Stamatatos, 2017). Neural methods such as Embedding, RNN, and CNN-RNN used GloVe word embedding (Pennington et al., 2014), but Stacked_CNN and Parallel_CNN did not use GloVe due to its negative impact on performance.

Table 7 summarizes the performance results. Surprisingly, the overall winner is Random Forest, outperforming all five neural models and four existing AA methods. As to per-class F1 scores, Random Forest, a robust non-linear model, accurately solved the AA problem across all nine test sets (one human and eight neural generators). Most generated texts were relatively easy to identify their authorship, giving up high F1 scores (especially the generators such as CTRL, GPT, XLM, XLNET, and PPLM).

The most challenging test set turns out to be both Human and GROVER that yields relatively low average F1 scores across all of classical, neural, and existing AA models (0.5423 and 0.5542, respectively). Also, interestingly, neural classifiers are able to classify FAIR very accurately unlike classical or existing AA models, while classical

Model	Balanced (1:1)			Imbalanced (1:8)		
	P	R	F1	P	R	F1
Embedding	0.4922	0.4877	0.4899	0.4555	0.5274	0.4770
RNN	0.7625	0.7611	0.7611	0.8242	0.6956	0.7390
Stacked_CNN	0.7592	0.7592	0.7592	0.6585	0.7252	0.6816
Parallel_CNN	0.9125	0.9118	0.9120	0.8370	0.8458	0.8413
CNN-RNN	0.7314	0.7315	0.7314	0.8198	0.7162	0.7546
RoBERTa	0.4949	**0.9540**	0.6517	0.1090	**0.9540**	0.1957
RoBERTa-tuned	**0.9196**	0.9109	**0.9152**	**0.9229**	0.7859	**0.8489**
GROVER-DETECT	0.8100	0.5590	0.6610	0.3337	0.5591	0.4180

Table 6: P2: Binary classification performance of "Human vs. Machine" on two collective test sets.

Model	Human	Machine								AVG
		CTRL	GPT	GPT2	GROVER	XLM	XLNET	PPLM	FAIR	
Naive Bayes	0.4668	0.9812	0.9835	0.4830	0.1901	0.9858	0.9810	0.9448	0.1812	0.6886
Decision Tree	0.7376	0.9835	0.9696	0.7239	0.6682	0.9837	0.9858	0.9626	0.5770	0.8435
SVM	0.8038	0.9953	0.9953	**0.8048**	0.7426	0.9953	**0.9976**	0.9742	0.6792	0.8876
Random Forest	**0.8122**	**1.0**	0.9953	0.7850	**0.8169**	**1.0**	0.9906	**0.9860**	0.7465	**0.9042**
Embedding	0.5727	0.9581	0.9688	0.7785	0.1080	0.9589	0.9026	0.7424	0.9900	0.7756
RNN	0.4190	0.9932	0.9906	0.7659	0.6295	0.9953	0.9929	0.8238	**1.0**	0.8456
Stacked_CNN	0.3415	0.9518	0.9638	0.7511	0.6603	0.9662	0.9104	0.8009	0.9950	0.8157
Parallel_CNN	0.5020	0.9790	0.9638	0.7579	0.6499	0.9976	0.9953	0.7582	**1.0**	0.8448
CNN-RNN	0.6366	0.9730	**1.0**	0.8038	0.5664	0.9813	0.9739	0.7942	**1.0**	0.8589
POS+CNN-LSTM	0.5868	0.6777	0.9109	0.7132	0.4798	0.8910	0.6845	0.8467	0.5689	0.7066
POS+LSTM-LSTM	0.2378	0.6746	0.8654	0.6512	0.4628	0.7572	0.6505	0.7520	0.5876	0.6266
3-grams + SVM	0.6992	**1.0**	**1.0**	0.6821	0.6579	**1.0**	0.9929	0.8165	0.6483	0.8330
Character n-gram + SVM	0.7008	**1.0**	**1.0**	0.6835	0.6534	**1.0**	0.9929	0.8114	0.6410	0.8314
AVG	**0.5423**	0.9360	0.9698	0.7218	0.5542	0.9633	0.9270	0.8366	0.7396	

Table 7: P3: multi-class classification performance with per-class macro F1 (for each column) and overall average F1 scores of models (for each row).

models, especially Random Forest and SVM, perform better for tough test sets such as GROVER and Human.

9 Discussion

9.1 P1: Same Method or Not

As expected, we find that the balanced setting yields significantly higher F1 scores across five neural models than the imbalanced setting. However, P1 is still nontrivial to solve, especially in the imbalanced setting, as can be seen in Figure 1, where many machine-generated texts are shown to be linearly inseparable. Furthermore, from Table 3 and Section 4, we can see that while some generators generate similar texts, all generated texts still possess distinct qualities that are leveraged in P1, achieving F1=0.9813 in the balanced setting. It is harder to grasp these distinct characteristics when looking at a single piece of text. As such, the comparison of two texts in the setting of P1 offers an advantage to the task.

9.2 P2: Human vs. Machine

We find that RoBERTa-tuned often outperforms neural classifiers in the individual human vs. machine setting, except for the case of GROVER (Table 5). RoBERTa-tuned outperforms all competing models in distinguishing machine texts from human texts, incredibly well on GPT2 texts (achieving F1=0.9792), probably due to sufficient training on GPT2 data. Next, we find that GROVER-DETECT underperforms in classifying the other machine-generated texts in Table 5, but performs well on Human vs. GROVER achieving the F1 score of 0.8135. This is because it was trained to detect GROVER-generated texts. For the collective settings, however, both RoBERTa and Parallel_CNN have similar F1 scores, while outperforming the rest by significant margins.

9.3 P3: Authorship Attribution

For this setting, in Table 7, we compare different settings, including (1) the use of GloVe word embedding with Embedding, RNN, and CNN-RNN; (2) no word embedding with Parallel_CNN and Stacked_CNN; (3) the use of linguistic features with classical learning algorithms; and (4) n-grams and POS-tags with state-of-the-art AA methods. In this task, we learn that the more accessible generators to classify are CTRL, XLM, and XLNET, while the harder ones are Human, GROVER, FAIR, and GPT2. This can be seen in Tables 5 and 3, where the more demanding generators underperform, and score highly in *LIWC-Analytic* and *LIWC-Article*, respectively. This is vice versa for the more accessible generator. We also find that the linguistic features effectively solve P3, slightly better than state-of-the-art AA solutions, and (simple) neural classifiers. The top stylistic features are *word count, article, period, word-per-sentence count, auxiliary verb, preposition, comma*. We expect this result will change in the future when: (1) the quality of machine-generated texts improve, losing revealing linguistic cues, and (2) neural models are trained better with an enormous amount of data and more powerful architectures.

One may wonder if some results with high F1 scores to solve P3 in Table 7 are simply due to the fact that different generators tend to generate texts on different topics (with non-overlapping word usage, thereby affecting embedding to neural models). In addition, while we only attempt to collect our articles from the domain of "politics," some other domains may have been added unintentionally. However, when we solve P3 using the combination of bigram and trigram models with top-20 LDA-extracted topics, we achieve only 0.38 as the overall average F1 score. Therefore, we believe that simple topical analysis of generated texts cannot solve P3 well.

10 Conclusion

We have conducted comprehensive experiments on three versions of the Authorship Attribution (AA) problem: (1) the same method or not, (2) human vs. machine (Turing Test), and (3) who is the author. Notable findings from our empirical evaluation include: (1) not all neural text generation methods generate high-quality human-mimicking texts–in particular, GPT2, GROVER, and FAIR generated better-quality texts and (2) using specific linguistic features and simple neural architectures, we can solve three problems reasonably well, except GPT2 and FAIR in P2 and GROVER in P3.

11 Acknowledgement

This work was in part supported by NSF awards #1742702, #1820609, #1909702, #1915801, and #1934782.

References

Hanan Alshaher and Jinsheng Xu. 2020. A new term weight scheme and ensemble technique for authorship identification. In *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis*, pages 123–130.

Bander Alsulami, Edwin Dauber, Richard Harang, Spiros Mancoridis, and Rachel Greenstadt. 2017. Source code authorship attribution using long short-term memory based networks. In *European Symposium on Research in Computer Security*, pages 65–82. Springer.

Grzegorz Baron. 2014. Influence of data discretization on efficiency of bayesian classifier for authorship attribution. *Procedia Computer Science*, 35:1112–1121.

Dainis Boumber, Yifan Zhang, and Arjun Mukherjee. 2018. Experiments with convolutional neural networks for multi-label authorship attribution. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164.*

Elisa Ferracane, Su Wang, and Raymond Mooney. 2017. Leveraging discourse information effectively for authorship attribution. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 584–593.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043.*

Oren Halvani, Lukas Graner, Roey Regev, and Philipp Marquardt. 2020. An improved topic masking technique for authorship analysis. *arXiv preprint arXiv:2005.06605.*

Julian Hitschler, Esther van den Berg, and Ines Rehbein. 2017. Authorship attribution with convolutional neural networks and pos-eliding. In *Proceedings of the Workshop on Stylistic Variation*, pages 53–58.

Armin Hoenen and Niko Schenk. 2018. Knowing the author by the company his words keep. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Renkui Hou and Chu-Ren Huang. 2017. Stylometric studies based on tone and word length motifs. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 56–63.

Fatma Howedi and Masnizah Mohd. 2014. Text classification for authorship attribution using naive bayes classifier with limited training data. *Computer Engineering and Intelligent Systems*, 5(4):48–56.

Fereshteh Jafariakinabad, Sansiri Tarnpradab, and Kien A Hua. 2019. Syntactic recurrent neural network for authorship attribution. *arXiv preprint arXiv:1902.09723*.

Patrick Juola. 2012. Detecting stylistic deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 91–96. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Mike Kestemont. 2014. Function words in authorship attribution. from black magic to theory? In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 59–66.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1566–1576.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

A. Pastor López-Monroy, Fabio A Gonzalez, and Thamar Solorio. 2020. Early author profiling on twitter using profile features with multi-resolution. *Expert Systems with Applications*, 140:112909.

Enrique Manjavacas, Jeroen De Gussem, Walter Daelemans, and Mike Kestemont. 2017. Assessing the stylistic properties of neurally generated text in authorship attribution. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 116–125.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. *arXiv preprint arXiv:1907.06616*.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Thomas Proisl, Stefan Evert, Fotis Jannidis, Christof Schöch, Leonard Konle, and Steffen Pielström. 2018. Delta vs. n-gram tracing: Evaluating the robustness of authorship attribution methods. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Yafeng Ren and Donghong Ji. 2017. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385:213–224.

Javier Sánchez-Junquera, Luis Villaseñor-Pineda, Manuel Montes-y Gómez, Paolo Rosso, and Efstathios Stamatatos. 2020. Masking domain-specific information for cross-domain deception detection. *Pattern Recognition Letters*, 135:122–130.

Yunita Sari, Mark Stevenson, and Andreas Vlachos. 2018. Topic or style? exploring the most useful features for authorship attribution. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 343–353.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273.

Thomas Schürmann and Peter Grassberger. 1996. Entropy estimation of symbol sequences. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6(3):414–427.

Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2012. Authorship attribution with author-aware topic models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 264–269. Association for Computational Linguistics.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2011. Authorship attribution with latent dirichlet allocation. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 181–189. Association for Computational Linguistics.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Computational Linguistics*, 40(2):269–310.

Jialin Shao, Adaku Uchendu, and Dongwon Lee. 2019. A reverse turing test for detecting machine-made texts. In *Proceedings of the 10th ACM Conference on Web Science*, pages 275–279.

Abhay Sharma, Ananya Nandan, and Reetika Ralhan. 2018. An investigation of supervised learning methods for authorship attribution in short hinglish texts using char & word n-grams. *arXiv preprint arXiv:1812.10281*.

Prasha Shrestha, Sebastian Sierra, Fabio A González, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674.

Lucy Simko, Luke Zettlemoyer, and Tadayoshi Kohno. 2018. Recognizing and imitating programmer style: Adversaries in program authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2018(1):127–144.

Thamar Solorio, Sangita Pillay, Sindhu Raghavan, and Manuel Montes. 2011. Modality specific meta features for authorship attribution in web forum posts. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 156–164.

Efstathios Stamatatos. 2017. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1138–1149.

Kalaivani Sundararajan and Damon Woodard. 2018. What represents "style" in authorship attribution? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2814–2822.

Michael Tschuggnall and Günther Specht. 2014. Enhancing authorship attribution by utilizing syntax tree profiles. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 195–199.

Adaku Uchendu, Jeffery Cao, Qiaozhi Wang, Bo Luo, and Dongwon Lee. 2019. Characterizing man-made vs. machine-made chatbot dialogs. In *Proceedings of the Int'l Conf. on Truth and Trust Online (TTO)*.

Andreas Van Cranenburgh. 2012. Literary authorship attribution with phrase-structure fragments. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 59–63.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Anelka Zečević. 2011. N-gram based text classification according to authorship. In *Proceedings of the Second Student Research Workshop associated with RANLP 2011*, pages 145–149.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9051–9062.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3):378–393.

Balanced	#train	#valid	#test
P1	68,896	7,656	19,139
P2	2,985	426	853
P3	6,825	881	1,888

Imbalanced	#train	#valid	#test
P1	62,157	6,907	17,266
P2	6,825	881	1,888

Table 8: Details on Train, Validation and Test Set Splits

A Reproducibility

A.1 Implementation, Infrastructure, Software, and Data

We run all experiments using either P100 or Titan Xp GPU card on a standard server machine with 16GB of RAM. We utilize deep learning platform *Ludwig* (v.0.2.1) with *Tensorflow* (v.1.15.0) backend to develop and evaluate all text classification models in the paper. For classical ML, we utilize *scikit-learn* (v.0.22.1) library. All implementations are done using *python* language (v.3.0). For generating text, we adopt various models implementation provided by *huggingface*[11], *PPLM*[12], *grover*[13], and *Fairseq*[14] Github repo. To extract LIWC features, we utilize the LIWC2015 software (v.1.6.0)[15].

A.2 Data and Preprocessing

We generate all the text following the description in Section 3. Since the generated text of some machine algorithms includes artificial tokens such as <eos> and <sos>, we remove these tokens from the results. We also ensure that the prompts (i.e., article titles) are appended to every generated article. For P3, we use all the generated text by 9 methods (human and eight machine algorithms), resulted in a dataset with balanced label distribution. For P1, creating datasets generators' pairs is a combinatorial problem, which will create a very large dataset. Instead, we sample from each possible pairs of generators K samples while maintaining the relative distribution among them, resulting in the imbalanced dataset. Then, we adjust K and under-sample negative samples with 1:8 ratio to create the balanced dataset for P1. For P2, we curated the imbalanced dataset from P3, with 1 human and 8 machine generators. Then, we under-sample negative sample with 1:8 ratio to create a balanced dataset for P2. For each task P1, P2 and P3, we then split to train, validation and test set with 7:1:2 ratio. Table 8 summarizes statistics of datasets used for each task in balanced and imbalanced scenario, respectively. Also, using language_check, a python package for detecting and correcting grammatical errors, we found that most generators had less than a 3% grammatical error rate, except for XLM that had a 14% error rate.

A.3 Running Time

All experiments take an average running time of around 2 minutes for each training epoch. Depending on the text encoders being utilized, and one training epoch can take as low as 10 seconds (Embedding model) to as long as 8 minutes (CNN-RNN model).

A.4 Training and Model's Parameters

For each of neural network models tested in the paper, we use various text encoders to learn vector representations of input texts (Section 5), results of which are then input into a fully connected network (FCN) with Dropout followed by a softmax layer to make prediction. Table 9 describes the training hyper-parameters and various models' architectures. We train all neural network models using Adam optimizer (Kingma and Ba, 2014) with default parameters.

Parameter	Value
Max Words	500
Vocabulary Size	20,000
Early Stop	2
Batch Size	256
Learning Rate	0.01
Adam Optimizer	β_1: 0.9, β_2: 0.999, ϵ: 1e-08
Embedding Size	300
Stacked CNN Kernel Sizes	7, 7, 3, 3, 3 and 3
Stacked CNN Pool Sizes	3, 3, 3, 3, 3, and 3
Parallel CNN Kernel Sizes	2, 3, 4 and 5
RNN Hidden Size	256
FCN Layers (before Softmax)	256 - 256
Dropout	0.5

Table 9: Model's Parameters and Training's Hyper-Parameters

A.5 Evaluation Metrics

We use standard *Precision (P)*, *Recall (R)*, and *F1* score as the main evaluation metrics throughout the paper. We first construct a confusion matrix and calculate those scores as follows.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = 2\frac{P * R}{P + R}$$

where TP is True Positive, FP is False Positive, FP is False Positive and FN is False Negative predictions.

[11] https://github.com/huggingface

[12] https://github.com/uber-research/PPLM

[13] https://github.com/rowanz/grover

[14] https://github.com/pytorch/fairseq

[15] https://liwc.wpengine.com