# Proactive Defense through Automated Attack Generation: A Multi-pronged Study of Generated Deceptive Content

by
Avisha Das

A Dissertation presented to the Faculty of Department of Computer Science,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Computer Science

Chair of Committee: Dr. Rakesh M. Verma

Committee Member: Dr. Ernst Leiss

Committee Member: Dr. Thamar Solorio

Committee Member: Dr. Pablo Gervás

University of Houston
December 2020

## DEDICATION/EPIGRAPH

This dissertation is dedicated to my parents, Dr. Mihir Kumar Das and Smita Das, for their unyielding love and unrelenting support. And to the memory of my beloved grandparents, Santosh Kumar Datta and Gita Datta for their love and blessings.

## ACKNOWLEDGMENTS

"*Life is simply a collection of memories, but memories are like star light... they live on forever.*" My life as a researcher has been an amalgamation of beautiful memories and moments, fortunate and unfortunate. This dissertation is my opportunity to thank the many faces who have been a huge part of my life, and are responsible for some of these moments.

I would first like to thank my advisor, Dr. Rakesh M. Verma, who has guided, encouraged, taught, supported and helped me in every phase of my graduate studies. His contribution to my research is immeasurable and his expertise was invaluable in enriching my research acumen. As an international graduate student in a new country and far away from home, I could not have asked for a better guide and teacher. Thank you so much, Dr. Verma, for being supportive and for believing in me. This work was supported in part by NSF grants DGE 1433817, CCF 1950297, and in part by the U.S. Army Research Office under contracts W911NF-16-1-0422 and W911NF-20-1-0254. I would also like to thank my committee members Dr. Thamar Solorio, Dr. Ernst Leiss, and Dr. Pablo Gervás for their support and patience.

I am so grateful to have been a part of the Reasoning and Data Analytics for Security (ReDAS) Lab at University of Houston. I have had the opportunity to come in contact with and learn from many people over the course of my graduate studies. Thank you to Vasanthi Vuppuluri and Nirmala Rai for your continued support and words of wisdom. I would like to take this opportunity to thank Daniel Lee, Shahryar Baki, Luis DeMoraes, and Samaneh Karimi – who have helped me immensely over the course of my Ph.D. I would also like to thank the current and past members of ReDAS Lab for their continued support and good wishes.

I have been lucky to have made friends who have become my family away from home. I would like to thank my friends, Dipanwita, Sriparna, Prajakta, Deepjyoti, and Sriharsha for being there and supporting me. Last but not the least, a very special thank you to Barry, Malpooa, and Arrow – you guys were, are and will be an integral part of my life. Finally, I would like to thank my parents and my family who have always been there for me as a constant source of support, love, and encouragement. I am immensely grateful to them for what I am today.

## ABSTRACT

Social engineering attacks are a security threat – attacks like phishing, email masquerading, etc. are common examples of such attacks where a perpetrator impersonates a legitimate entity to steal an unknowing victim's digital identity. However, despite having a higher probability of success, executing such an attack can be costly in terms of time and manual labor. With the advancements in machine learning and natural language processing techniques, the attackers can now use more sophisticated methods to evade detection. Deep neural learners are capable of natural text generation when trained on huge amounts of written textual content.

While these techniques have been tested in creative content (stories) generation based tasks, they have been abused to generate fake content (fake news) as well. In a proactive scenario, the defender presumes that attackers would resort to sophisticated yet automated methods of attack vector generation. However, the application of neural text generation methods to email generation is fairly challenging owing to the presence of noise or sparsity in emails and the diversity in email writing style. Moreover, the evaluation and detection of generated content is a challenging and cumbersome task and current automated metrics do not provide the best possible alternative.

We analyze the task of automated content generation for two tasks: (a) creative content or story generation from writing prompts; and (b) generation of emails from given subject prompts for specific intents. We split the proposed analysis for each task into three defined parts – (i) content (story/email) generation; (ii) fine-tuning and improving upon generated content; and (iii) content evaluation. Apart from testing the baselines like word-based Recurrent Neural Networks and pre-trained and fine-tuned transformer language models, we propose HiGen – a hierarchical architecture that leverages the use of a generative language model by improving upon the generated content with the use of sentence embeddings given a prior conditioning prompt. Finally, we compare the linguistic quality of the generated text to human authored text using a set of automated metrics. We also corroborate our findings with a human-based user study – to ascertain how well the metrics can distinguish between writing patterns. Moreover, we explore if there exists a difference in system performance with respect to the genre of text generation – story vs. emails. We see the overall improvement in sentence coherence in content generated by HiGen architecture.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# 1 Introduction

Adversarial learning is a major threat to the field of computer security research. With the advancement in technology, the growing dependency on the Internet has exposed users to serious cyberthreats like phishing and pharming. Despite considerable research to counter such threats, staggering numbers of individuals and organizations fall prey to targeted social engineering attacks incurring huge financial losses.

Although attackers change their strategies, previous research [13] has shown that electronic mails (emails) are a popular form of attack vector. More than 250 billion emails are sent each day[1] worldwide, making emails the major mode of communication and a largely preferred target vector by perpetrators. Emails can be embedded with a variety of malignant elements [24] like poisoned URLs to malicious websites, malware attachments as well as executables, documents, image files, etc. The Anti-Phishing Working Group (APWG) reports over 132,553[2] unique phishing email campaigns received in the $4^{th}$ quarter of 2019, rising from a total of around 122,359[3] unique reports identified in the $3^{rd}$ quarter of 2019. In 2020, 146,994 unique phishing sites were reported in the $2^{nd}$ quarter of 2020.[4] Verizon's 2019 Data Breach Investigation Reports[5] reveal that 32% of the confirmed data breaches involved phishing. Phishing reports also reveal the consistent rise in phishing attacks targeted towards financial institutions like payment processing firms and banking sectors with an estimated loss of 26 billion dollars globally[6] to phishing attacks. The statistics demonstrate how the threat continues getting serious and worse as attackers continue to devise more sophisticated (and maybe more effective) ways of scamming victims.

Innovative and unseen attack vectors can trick pre-trained classification techniques [86, 108], thus placing the victim at risk. **Email Masquerading** is an advanced and more targeted form of social engineering attack. The masquerader or attacker first gains access to the email account of an

[1] https://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf

[2] http://docs.apwg.org/reports/apwg_trends_report_q4_2019.pdf

[3] http://docs.apwg.org/reports/apwg_trends_report_q4_2019.pdf

[4] https://apwg.org/trendsreports/

[5] https://enterprise.verizon.com/resources/reports/dbir/

[6] https://www.ic3.gov/media/2019/190910.aspx

individual (we call him/her the *target*), thus compromising the digital identity of the target. The attacker then uses the target's credentials to compose and send the emails. The perpetrator can carefully construct a fraudulent email with poisoned links and/or attachments, which he/she can send out to the target's contacts (the *victims*). This has serious implications, because the attacker has gained uninterrupted access to the inbox, outbox and other private details of the compromised person. Thus, by exercising caution, he/she can emulate the content and context of the emails written by the individual and can communicate with the target's contacts as a legitimate entity, successfully evading detection and causing harm to the victim.

However, construction of the perfect deceptive email requires fine-tuning and manual supervision. Figure 1 gives an overview of this process. While a fake email constructed manually by an attacker can guarantee a higher chance of success, the process is both time and labor intensive. In contrast, an automated text generator can be trained to synthesize targeted emails much faster and in bulk, thereby increasing the chances of a successful attack, as shown in Figure 2.

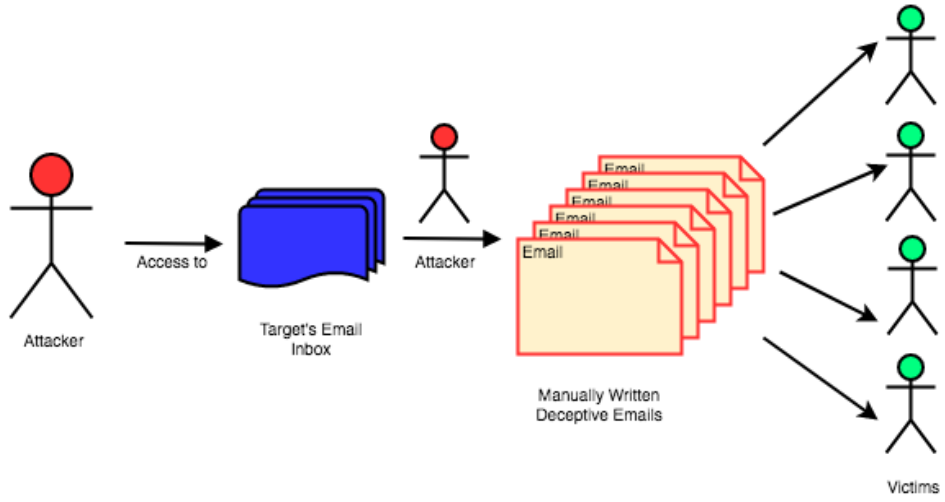



Figure 1: Email masquerading done manually

Reports[7] reveal that nearly 86% of the email-based attacks are "malwareless". This statistic demonstrates the increasing seriousness of the targeted spearphishing based attacks. However, the bottleneck in this case lies in whether the system can generate high quality text, free from common

[7]https://content.fireeye.com/one-email/ig-the-3-ts-of-email-attacks

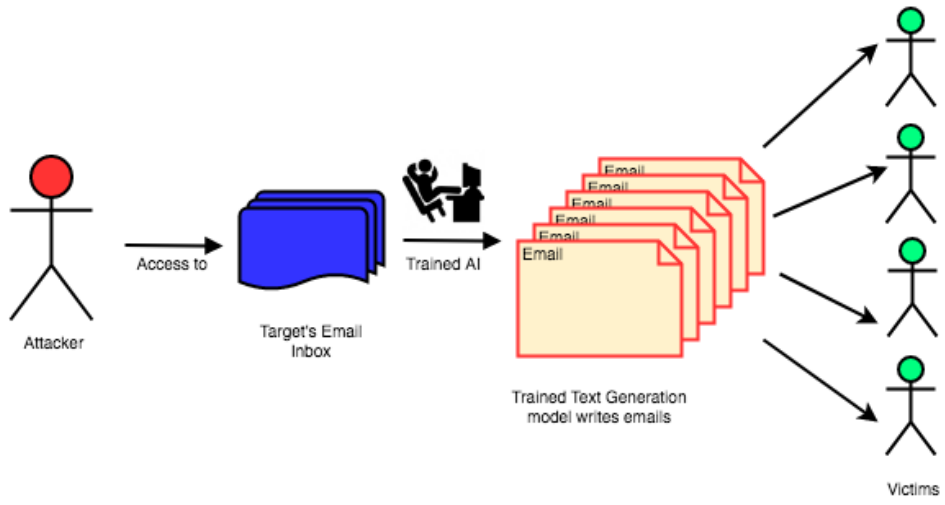



Figure 2: Email masquerading with trained AI

malicious flags like misspellings, incorrect and abusive language, over-usage of action verbs, etc., which can be picked up by a classifier easily. Can an automated generative system be trained to generate textual content emulating the stylistic features of human writing?

Proactive research in deception based attacks using email masquerading techniques is limited. Moreover, to reinforce existing email-based attack detection filters against newer and advanced attacks, further sophisticated experimentation is necessary. Advances in the field of natural language processing have introduced newer and sophisticated algorithms that enable a machine to learn and generate high-quality textual content on a given context. Grammar based tools like the Dada Engine [7], N-gram language models [16] as well as deep neural learners [108, 21, 116] have been used to study and replicate natural language based attacks. The aim is to facilitate proactive security by predicting newer attacks and reinforcing against such unseen yet impending threats.

In the attacker's hands, language generation techniques can become dangerous tools for deception. With access to proper training data, deep learning neural networks can generate textual content. This property has been leveraged by researchers for a wide variety of natural language generation tasks like tweets [85] and poetry [34], [106]. While limited research has been pursued using deep learners for generation of fake reviews [108], fake news [116], grammar based techniques [7] as well as simplistic deep networks [21] have been leveraged for email generation. Thus, we can

assume that it is not long before phishers and even spammers resort to such techniques to generate newer kinds of malicious attack vectors.

Natural language generation has gained popularity with the ready availability of language resources and language models, which can be used to emulate the stylistic aspects of the texts on which they are trained, ranging from fields of computational creativity to the generation of fake writing. An integral part of such automated generation systems is a trained language model that can be used to emulate the stylistic aspects of the text on which the model was trained. These models can be fine-tuned to model the writing style of single or multiple authors and to generate written content very similar to their human counterparts. Beyond generating textual content such as stories and poems, language generation systems have seen many diverse applications, for example, in conversational dialogue systems [46], automated headline generation [32].

With the feasibility of deep neural architectures, learning models of large textual content has become popular. Researchers have proposed several architectures capable of learning robust representations of natural language. These include encoders-decoders [27], generative adversarial networks [28], and transformers [95]. In recent years, the use of large-scale neural language models such as OpenAI's GPT [73], AllenNLP's ELMo [68], Google's BERT [22], Google/CMU's XLNet [107] has improved performance on natural language understanding tasks considerably. With the growing prominence of deep learning, an approach known as *end-to-end learning* [27] has become popular. Researchers have proposed several novel architectures capable of modeling robust representations of natural language – recurrent neural networks (RNNs) [90], sequential encoders-decoders (sequence-to-sequence learning) [27], generative adversarial networks (GANs) [28], and transformers with attention-modeling [95]. These pre-trained language models can be largely misused by attackers and perpetrators to generate fake content. This is a security threat and may hamper a unknowing victim.

To test the potential of these massive language models (LMs) in the realm of open ended content generation, we study the reproducibility and generalizability of these LMs in generating stories from prompts. We study the behavior of these pre-trained models in a *"zero-shot" setting*

and with *fine-tuning* on a dataset of human written stories and writing prompts. We also recognize that a selection of sampling parameters (for example, temperature, top-$k$ value, etc.) for generation play an important role in determining the quality of the text generated. To evaluate the generated content, we use a range of metrics to compare with human written references – semantic relatedness, linguistic quality and syntactic style.

This study of the deep generative transformer-based models explores the answers to three primary questions – (a) Are large language models comparable in performance to human writing with or without fine-tuning? (b) Are there specific combinations of sampling parameters (like temperature and top-k) at which the model generates the best samples? (c) Are some quantitative metrics better at identifying linguistic quality in content generated by machine vs. human writing? Thus, before applying these models to the realm of email generation, we identify the top pre-trained and transformer-based language models performing the best with respect to open ended content generation using story prompts.

Extracting the knowledge about the best language models and sampling conditions, we apply these models to the generation of legitimate looking emails and email threads. We propose a systematic setup in building and testing a proactive defense strategy. The proactive mode of study begins with a review of how existing baseline Recurrent Neural Network (RNN)-based architectures can be trained and used to model humans writing emails. The RNN-based language models were popular systems for text generation in the pre-transformer era, previously used by researchers [106, 34, 108]. We systematically improve upon the baseline word-based prediction model by including a hierarchical sentence selection model, that selects more semantically accurate sentences based on sentence-level embeddings. However, such a hierarchical setup generates text suffering from incoherence and repetitiveness.

On the other hand, transformer networks have been shown to generate long pieces of coherent text given a particular topic or seed. Thus, in this dissertation we leverage the use of transformer language models for automated generation of emails by improving upon the proposed hierarchical

model with a ranking and feedback loop. We visualize email generation as an open-ended generation task and apply the lessons learnt from a prior study of massive LMs in story generation to automatically write emails.

This helps us visualize how and to what extent these pre-trained models can be exploited by attackers in a practical setting. While we plan to identify the underlying implications of how an automated machine learning technique, here, deep learners can be leveraged to synthesize email bodies for the purpose of email masquerading attacks, we also study how combination of lexical features and pre-trained language models in a hierarchical setting can help better model the generated textual content.

We also demonstrate the systems' performance using qualitative and quantitative methods. First, we examine the reproducibility and generalizability of multiple massively pre-trained language models in the realm of natural language generation in a zero-shot setting. Then we fine-tune the pre-trained language models on legitimate email datasets to capture the patterns in which legitimate emails are written. Though anecdotal evidence suggest that these large language models generate high quality text, there is a limited analysis of the quality of the generated content. We collect a wide variety of automated evaluation metrics proposed in previous research as well as propose a novel set of sentence level evaluation metrics which can be used to extensively study the nature and the quality of the generated content. These metrics can be further extended to either detect fake content or to improve upon proposed text generation models.

### 1.1 Key Contributions

We summarize the key contributions as follows:

- Through an in-depth analysis of different transformer-based and recurrent neural models (Sections 4 and 5), we empirically find which combination of sampling hyperparameters perform the best in a zero-shot as well as fine-tuned setting. We observe that higher softmax temperatures (0.75, 1.0) and $k$ sampling values more than 50 (i.e., 150, 250, 500) are the best conducive to generating samples closest to human writing.

- To identify the best email subjects to be used as priming sequences for generation, we manually annotate 1000+ emails from the Avocado corpus and use a pre-annotated Enron corpus. Along with a category-based frequency distribution of the email intents and lexical analysis of the linguistic quality of the email bodies, we select *delivery of information* category as the most frequent as well as acceptable ranges of values for the lexical attributes.

- We propose a hierarchical sentence-level architecture for text generation in Section 6.1 – we leverage the trained generative language models (pre-/transformer) for generating novel sentence candidates with the sentence level coherence being controlled through comparison with sentence-based neural embeddings (Doc2Vec and S-BERT). Unlike the existing generative SOTA models (GPT, GPT2), the proposed architecture takes into account a larger range of sampling hyperparameters during generation of sentence candidates. Moreover, the addition of a novel neural sentence embedding model guides in selecting the best sentence candidates from the generated instances.

- We propose a measure of coherence at a sentence level – the harmonic mean of statistical properties of sentence connectedness in Section 3.5. The metric takes into account the relatedness across the sentences unlike the $n-$gram based metrics like BLEU and Rouge. Apart from measuring the coherence in the generated content, we also test the semantic quality and syntactic style by studying metrics like readability scores, POS tag frequencies, sentence length and prompt-based conditioning.

- We analyze the output of each architecture using the evaluation metrics mentioned and study the model performance depending on the nature of the training input (story or emails), the selection of hyperparameters, and finally the type of model as shown in Sections 4, 5 and 6. Each architecture and generative language model has been evaluated on two genres of text – stories from provided story prompts (WritingPrompts) and emails from Enron [26] and Avocado [23] corpora for the open-ended generation.

- We perform an in-depth review of the metrics used to evaluate the nature of generated content

by studying the inter-metric correlation of the metric scores on the outputs generated by each architecture. This shows which metrics are more unique and which ones have strong correlations among themselves. In order to determine the performance of the evaluation metric, we propose a metric ranking algorithm based on the metrics' scores on human authored content and generated content using a regression based analysis. These metric-based analyses are shown in Sections 4.2, 5.2 and 6.4 for the Word-RNN, the transformer-based LMs and the proposed hierarchical architecture respectively.

- We analyze the results of the higher ranked metrics by varying the sampling hyperparameters (sampling parameter $k$, softmax temperature $\tau$) for language models. We apply the language models along with the proposed architecture to the task of open-ended generation – stories from provided story prompts (WritingPrompts) and emails from subjects from Enron and Avocado corpora. We then evaluate the generated content using the proposed metrics and select the best hyperparameters.

- We perform a qualitative analysis of the story and email samples generated at different combinations of sampling hyperparameters - temperature ($\tau$) and top-$k$. We also perform a task-based effectiveness analysis using a trained email detection classifier for each architecture. This is to study the effectiveness of the generative language model in generating emails which can evade detection. This is shown in Sections 4.2.3, 5.2.3 and 6.4.3.

- In this dissertation, we have compared in depth the performance of generative models with respect to the applications on open-ended or narrative content generation (story) and conversational content generation (email/email threads). We further compare the nature of the stories and the emails generated by the language models in Section 8.

## 1.2 Dissertation Organization

Section 2 briefly discusses the related literature in natural language and fake content generation as well as emails. It also gives an overview of malicious email detection methods along with

evaluation of generated content. Section 3 gives an overall background of the necessary preliminaries – encoding techniques, datasets used for training and evaluation, neural architectures and networks and generative sampling methods. Section 3.5 describes the novel and prevalent evaluation metrics useful for evaluating generated content in detail. We first study in detail the performance of the word-level RNN along with quantitative and qualitative evaluation in Section 4. An in-depth study of the pre-trained and fine-tuned transformer-based language models is done in Section 5. We decribe the details of the hierarchical generative architecture – HiGen in Section 6.1 followed by further quantitative and qualitative analysis. Finally, we perform a comparative evaluation of the two types of content generated – in terms of qualitative and quantitative metrics in Section 8. Finally, we discuss the conclusions and future avenues in Section 9.

### 1.3 Notations

We list the notations of the architectures and language models in the Box 1.1.

**Box 1.1: Notations for Neural Architectures and Language Models**

**(A) RNN: Recurrent Neural Networks**
*wRNN*: Word-based RNNs
*LSTM*: Long Short Term Memory Networks
*Bi-LSTM*: Bidirectional LSTMs

**(B) Transformer-based Networks**
**(I) OpenAI GPT Pre-trained Models:**
*OG:* openai-gpt (110 million); *G2:* gpt2 (117 million); *GM:* gpt2-medium (345 million);
*GL:* gpt2-large (774 million)
**(II) Google/CMU's Transformer-XL and XLNet Models:**
*XB:* xlnet-base-cased (110 million); *XL:* xlnet-large-cased (340 million);
*TX:* transfo-xl-wt103 (257 million)

**(C) Neural Sentence Embeddings**
**D2V:** Document2Vector or Doc2Vec Embeddings
**S-BERT:** Sentence Bidirectional Encoder Representations from Transformers (BERT)
**S2V:** Sentence2Vector Embeddings

# 2 Related Work

The need to counter the active attacker issue has given rise to proactive methods. While there exist classical techniques for phishing email detection [98], [4], [5], state-of-the-art malicious email detectors fail to detect a sophisticated or targeted attack like email masquerading [7]. This has inspired researchers to delve deeper into the realm of attack generation and experiment with newer and different attack vectors. Such works involve generation of news, reviews, tweets, etc. While the use of fully automated methods for text generation has been considered, there has not been much research into automatic modeling of coherent emails that can be used for targeted attacks.

In this section, we review the existing literature in natural language generation, generation of attacks and email generation. We also include research that review the detection of phishing or malicious emails and evaluate generated text quality and differentiate between generated text and human authored text. We start with a study of the common language generation architectures and then briefly review the works that have used neural learning to generate fake content or exploits. We then move to existing literature in phishing email detection, automated evaluation metrics, and works that compare their effectiveness to human judgement.

## 2.1 Textual Content Generation

Deep neural networks have enabled building fully (or partially, with feature engineering) automated models for natural language generation. From the perspective of written text, a substantially trained deep network can emulate the writing style of an individual [90]. This property has been leveraged in natural language research by making deep learners write a wide variety of text – Shakespearean Sonnets [106], poetry [34], [112], answer generation [60]. While the use of grammar [7], templates [16], [17] and statistical language based models (e.g., N-grams [36]) are popular, Recurrent Neural Networks (RNNs) have been shown to be a more suitable choice owing to their ability to learn dependencies across the textual context [37]. Long Short Term Memory (LSTM) networks are more suitable for longer text sequences. However, while a fully automated system

seems promising - controlling the coherency, topic and structure of the generated text can be quite challenging. We divide the literature into two separate sections based on the nature of generation.

### 2.1.1 Open-ended Content Generation

Unconditional or open-ended content generation systems do not adhere to given priming conditional statements. Moreover, the length of the content generated also plays an important role in the quality of the text generated. Therefore, we look separately at systems that generate shot- and long- form textual content.

**Short-form Textual Content.** The paper [7] uses a grammar-based approach to manually build rules or grammars for representing emails. These grammars are then passed onto the Dada engine to generate legitimate looking synthetic emails. The authors use Sarah Palin and Hilary Clinton's emails to compose their dataset of generated emails. Their evaluation with a user study of 34 participants revealed that 17% of the participants successfully identified malignant signals in the emails and the users' detection rate was close to random guessing. Das et al. [21] use a more automated deep neural network built using word-based LSTMs for email generation. The authors vary the percentage of the malicious (Nazario emails and personal phishing emails) and legitimate emails (WikiLeaks' Palin and Clinton emails and Enron emails) while being fed as input to the training model. However, the system suffers from lack of training data and the output instances in turn are incoherent and incongruous in context as shown by their qualitative evaluation. But the proposed system though simple also generates emails with malicious textual cues [24] as observed in phishing emails. Using advanced NLG techniques and tools, [35] describe Community Targeted Phishing (CTP) attacks along with flow charts demonstrating the construction of malicious emails.

Emails are the most common and preferred method for social engineering attacks. [24] describes the *modus operandi* and the structure of a common phishing email. Researchers have also delved deeper into the attributes and underlying psychological features that cause phishing and social engineering attacks to be successful in [30], [83], [35]. Techniques for automatic generation of synthetic emails have been discussed in [7], [16], but introducing attributes of deception into

legitimate emails is a non-trivial task [21], [14], [64].

**Long-form Textual Content.** Attack vectors synthesized with little manual post-processing increase the chances of the attack [116, 108]. In [108], the authors use an LSTM-based deep neural learner trained on reviews from Yelp consisting of fake and real instances. The neural generative model is character-based, and the authors apply post-processing techniques to further polish the hotel reviews to make the text more believable. The authors perform a user-study based survey to test the effectiveness of the generated reviews and propose detection mechanisms to detect the synthesized fake reviews. The issue of fake news generation using transformer based models like GPT2 was explored by the authors of [116]. They trained a transformer-based model similar to GPT2-117M on a news dataset of size 120GB consisting of real news articles collected by scraping the CommonCrawl.

Using RNNs for sequence-to-sequence learning is a popular practice for text generation as shown in [29], [117], [71], [51]. Since simple encoder-decoder architectures fail to model important or meaningful words and phrases, [53] and [29] use attention based encoder-decoder models for preserving coherence and context in the generated text. Other generation techniques include deep learning with Markov Models [104], variational auto-encoders [76], and generative adversarial networks [70]. While previous research experiment with LSTM-based architectures, such strategies are prone to generating incoherent content as the length of the generated text increases. In previous research, the use of simpler RNNs [63] has led to generation of incoherent text sequences [59, 36, 90]. Martin et al. [62] proposed an event based approach for text generation using a corpus of movie plot summaries extracted from Wikipedia. The system is an amalgamation of two separate architectures. The proposed character, word or even sentence level architectures suffer from issues of incoherence. The paper proposes the use of multi-layer encoder-decoder network to extract events from sentences in a story. The authors use techniques like named entity recognition, genre information as well as dependency parse information to extract events from sentences and create 4-tuple generalized event representation from each sentence. Each such event tuple has 4 elements – object, subject, verb and modifier. They found that the events that over-generalize the main context of the given input

contribute the most to the improvement of a generative recurrent neural network.

Other studies in an adversarial setting, that leverage natural language techniques, have been pursued in spreading malicious Twitter messages [82], generating malicious URLs [6], generation of fake reviews [108], fake comments in opposition to net neutrality in 2018 [39] as well as text messages [84]. Authors of [82] propose SNAP_R – a system built using recurrent neural network trained using spear phishing pen-testing data that has dynamically generates Tweets extracting topics from the timeline posts of targets and the users they retweet or follow. The model is augmented with a clustering module to identify more popular targets based on the degree of social engagement to measure the effectiveness of the attack. In [6], the authors explore AI based threat vectors by analyzing millions of phishing URLs. They leverage deep neural learners (DeepPhish algorithm) for two different attack vectors involving different phishing URL domains that improved the effectiveness of the proposed system.

### 2.1.2 Prompt Driven Generation

Conditional content generation leads to generation of text that is related in content to a given prompt or starting seed. For examples, authors of [27] proposes a Seq2Seq architecture that conditions on a given writing story prompt. Here, we describe the literature that propose systems generating content from given starting seeds based on the nature of the content generated.

**Short-form Textual Content.** In recent times, [12] proposed the GPT-3 language model - a scaled-up version of the GPT-2 models [73]. The GPT-3 language model proposed and evaluated in this paper has been trained with 175 Billion parameters. The authors test the system in a variety of settings (zero-shot, few-shot and fine-tuning) and on many tasks – natural language inference, reading comprehension, closed book QA, machine translation. While the authors evaluated the model and reviewed the performance on a variety of tasks, the application of these models to language generation is of the greatest importance to us. The authors perform a human study using 600 participants to assess the linguistic quality of synthetic news articles. Lower accuracy scores to distinguish the fake from the original human authored article, despite increased time

investment from participants supports the finding that larger models generate harder-to-distinguish news articles. The ability of the perpetrators to deceive an individual by behaving as a legitimate entity can be automated for widespread social engineering attacks as studied in [64] and [47]. Researchers in [7], [21], [35], review 'weaponizing' advanced machine learning techniques to launch sophisticated yet automated targeted attacks.

**Long-form Textual Content.** In [27], researchers proposed a state-of-the-art hierarchical neural fusion architecture using two seq2seq models [87] along with a multi-scale gated attention mechanism ensuring relatedness of generated content to a given prompt. The proposed hierarchical architecture in this paper has been used for open-ended story generation from prompts. Standard sequence-to-sequence models have found application in hierarchical story generation [91], but such architectures fail to generate content that relates to the given story prompt [27]. Li et al. [58] use an LSTM to hierarchically learn word, then sentence, then paragraph embeddings, then transform the paragraph embeddings into text. Authors in [109] generate a discrete latent variable based on the context, then generates text conditioned upon it. The pre-trained deep GPT2 language model [73] from OpenAI has gained a lot of attention in the realm of content generation [66]. The authors of [79] have compared the Fusion model [27] with the GPT2-117 model [73] in open ended story generation. For the decoding algorithm, the researchers of [79] use top-$k$ sampling method as proposed by [27, 73] instead of beam search or greedy sampling methods [32].

### 2.1.3 Automated Email Generation

Previous literature on automated email generation methods is limited due to the nature of the text. Emails are harder to generate, due to nature and 'anatomy' of an email. Also, the writer's intent as well as relationship with the receiver(s) play important roles in ascertaining the tone and content of the email.

Yu et al. [113] proposes a phishing email generation technique based on data insertion to address the problem of data availability and balancing features available in benign and malicious emails. They propose the generation of samples by insertion of elements from benign data. The

implemented method generates and inserts six kinds of HTML resources where the selection of the resource is based on two parameters: control sequence and quantity sequence. The control sequence is a set of True-False switches used to control whether a type of resource is inserted in the sample; the quantity sequence is a set of values, generated by a certain rule, and used to indicate the amount of each resource inserted into the mail. The proposed system in the paper generates control sequences and quantity sequences based on the principle of randomness (random value of the quantity sequence is between 0-50). With data preprocessing, the authors used 72,614 legitimate emails from Enron and 9,647 phishing emails from Nazario's phishing email campaign. The authors generated 28,196 phishing samples and 35,689 Enron samples. They extract 12 statistical features for evaluation from the dataset and RandomForest algorithm was used for classification. After 10-fold cross validation, the accuracy was 96.72%. The newly generated data were observed to optimize the accuracy rate of the model.

The paper [89] proposes a method for generating realistic emails for multi-agent simulations using a distributed model of email threads. The proposed model uses data-driven agent-based modeling for email generation. The first step of the approach consists of using template-based generation methods for composing emails. The second part of the approach leverages social network analysis based methods to build distributed communication graphs that can be used by agents. Finally, the authors propose methods to measure the fidelity of the output graphs to ensure if the changes made are realistic in nature. The authors use the sent emails from the Enron Email Corpus. The preprocessing step removes the quoted text and message header information. The template-based email generation [16, 17] consists of data preprocessing steps followed by topic modeling and named entity recognition in the dataset. For building the email templates, the authors use the above two methods to generate generic emails. The run-time generation method generates emails by topics and reusing information from a set of scenario-specific entities, as well as names for sender and recipients, and a desired email topic. The second part is thread generation and distributed thread model building that uses the communication between agents for generating emails based on agents' needs. To evaluate the approach, the authors use 1000 generated email

threads and randomly select 1000 human-authored email threads to create two evaluation corpora. The authors measure the degree centrality and mutuality scores of the graphs for each evaluation corpus.

## 2.2 Phishing Email Detection

Researchers in [5] extract features from email's body, URL, and HTML content, used by supervised (SVMs, Neural Networks) and unsupervised (K-Means clustering) detectors to predict email nature. The system in [4] extracts 25 stylistic and structural features from emails, used by an SVM-based detection filter. Newer phishing email detection techniques are based on textual content analysis like the algorithms proposed in [97, 100, 98, 99, 114]. Researchers also leverage semantics to increase the robustness of the features and hence the performance of classifiers like the use of WordNet[8] to enrich the textual features in the papers [100, 97, 80, 96, 110].

Verma et al. [97] use natural language processing techniques to identify phishing emails from benign counterparts. The proposed detector, PhishNet-NLP parses an incoming email into its components - header, links and body text and each part is separately analyzed by a classifier. Finally, the scores from the three classifiers are combined to provide a final judgement on the nature of the email under analysis. The paper [100] also leverages information acquired from the semantics of an email and combines that with statistical features to propose a system for phishing email detection.

In a real-world case, there exists an imbalance between the malicious and legitimate emails with phishing emails being much lower than legitimate ones. The papers [41] and [115] used imbalanced datasets made of legitimate and phishing emails from SpamAssassin and Nazario respectively to build their phishing detection system. An instance of a realistic ratio is [11], where the authors collected 36,364 ham emails and 3,636 phishing emails from private sources. This research addresses the issue of data availability for training and evaluation purposes.

Considering the ever-evolving nature of phishing attacks, using old sources to train and evaluate

[8] `https://wordnet.princeton.edu/`

classifiers is an issue. Companies can use their private email logs collected from recent months or years for their research, like [111] gathered phishing emails from an Australian bank in the span of 5 months and [1] where the authors used their own private server to collect legitimate and phishing instances. However, the use of private datasets exacerbates the problem of comparing systems: researchers do not have access to such proprietary datasets and therefore must resort to older email samples to test their systems. With automated methods of email generation, researchers can train their systems to detect fake or synthetic emails thus building better and advanced solutions.

Researchers test for robustness in different ways. One way would be to train the phishing email detector on one subset of the dataset, but test it on another subset that was collected in a later time span, e.g., [11]'s system achieved an $F_1$-score of 98.66% on test dataset compared to the original $F_1$-score of 99.89% on the training data. Another way to test for robustness and how the classifier would perform against zero-day attacks is to train the model on one dataset and test it on emails from a different dataset. In [1], classifiers trained on Nazario and SpamAssassin datasets were tested on emails collected from private servers; recording a performance drop from 99% to 98%. Legitimate datasets from Enron were varied in evaluating the systems for [100] – trained on a subset of inbox emails from Enron and then tested on a different subset of inbox and outbox emails from Enron.

## 2.3 Evaluation of Generated Text

We organize the literature on content evaluation metrics based on syntactic and semantic properties.

### 2.3.1 Syntactic Evaluation

Bangalore et al. [8] proposed automated accuracy-based metrics, which account for string matching as well as matches in the dependency-based parse tree to quantify the level of agreement between a given reference and the generated content. They also *manually* rate the generated content for quality and understandability using a scale of numeric scores from 1 (lowest) to 7 (highest). The paper [43] contains a comparison of automated and subject-based approaches for synthetic text

with gold-standard reference texts using an NLG-based case study called ENIGMA. A Turing-style test for quality evaluation was also proposed in this paper. Several grammar-based metrics, count of misspelled words, parsing score, and percentage of word overlap (BLEU), were compared with human evaluation results in [65].

For evaluation of generated stories, Fan et al. [27] and See et al. [79] propose a set of story-prompt relatedness measures along with other metrics for repetition, rareness, and syntactic style and complexity of the generated text. Researchers also considered readability scores [65] (Flesch-Kincaid Reading Ease, Dale-Chall Readability Score, etc.) and constituency parse scores [40] to evaluate the readability and grammaticality of generated content. Lapata and Barzilay [56] propose automated metrics for coherence of machine generated summaries. To study the statistical differences between generated and human authored text, [33] proposed an evaluation and detection tool called 'GLTR'. The tool applies a collection of baseline statistical methods to detect generative text from human written content based on observations of word reuse rate, word ranking in the vocabulary, word probability and entropy. They also show that using their proposed tool as a guide improved human detection performance considerably. A unified framework evaluating both diversity and quality based on the optimal error rate of predicting whether a sentence is human-written or machine-generated. Using two tasks – summarization and dialogue generation – they evaluate a proposed metric HUSE that combines human and statistical evaluation.

### 2.3.2 Semantic Evaluation

Evaluating linguistic and semantic quality of generated text is essential. However, there could be a bias in choosing a metric or a method to evaluate the generated content automatically [94]. While existing automated evaluation metrics are not the best performers, manually evaluating generated text quality can be time consuming and prone to bias [65]. Authors put together a comprehensive evaluation of semantic-based automated metrics in [75]. To capture semantic relations across sentences at a word-level, the authors of [118] propose a similarity-based evaluation metric BERTScore. This metric is shown to perform better at measuring linguistic quality than existing metrics BLEU,

ITER by correlating the score with human level judgements on two natural language understanding tasks: image captioning and machine translation.

Recently, researchers have investigated methods that compare text quality while considering hyperparameters that influence sample decoding from trained deep language models. The authors in [49] investigate how automated discriminators compare with human evaluators in this context. They also explore whether factors like sampling temperature and decoding parameters play a role in controlling the nature of the generated content. The paper [49] comprehensively studies the discriminators of machine-generated text, and their sensitivity to model structure, sampling methods, and excerpt length. The authors analyze the human raters' ability to identify machine-generated content, and of how human decision-making differs from that of automatic systems.

Grammaticality and semantic correctness are two important attributes that can evaluate the quality of generated text. In [65] also reports results using a semantic similarity measure based on distributional similarity in text and Latent Semantic Analysis proposed by [42]. Researchers in [15] release a grammatical classification and semantic correctness classification dataset for the weather domain that consists of responses generated by three data-driven NLG systems. They also explore the use of two supervised learning approaches (CNNs and gradient boosted decision trees) for classifying grammaticality.

# 3 Background, Metrics and Preliminary Analysis

This section gives an overview of the datasets used in training, fine-tuning and evaluating the language models and deep neural architectures. We study the architectures and networks used in text generation and finally explain the sampling techniques used for generating the samples from the models. We also provide an in-depth review of the textual cues present in phishing emails and an analysis of the various intents existing in organizational emails.

## 3.1 Byte Pair Encoding

The Byte Pair Encoding (BPE) tokenization scheme was introduced in [81] as a data compression technique to improve machine translation tasks. The BPE tokenization scheme helps address the issue with out-of-vocabulary or rare words by using subword tokens. The main idea stems from the use of lossless data compression methods. The method reduces the total vocabulary size by keeping more frequent words while replacing the less frequent ones with a sequence of tokens. The BPE method ensures a balance between character- and word-level hybrid representations thus making it capable of effectively encoding large corpora. In this research, we use this tokenization scheme for the purpose of filtering the dataset and fine-tuning the language model.

The tokenization scheme has paved the pathway for building the wordpiece and sentencepiece tokenizers used in deep neural learners like BERT [22], GPT2 [73]. The scheme also enables the encoding of any rare words in the vocabulary with appropriate sub-word tokens without introducing any new or unknown tokens.

## 3.2 Datasets

Depending on the task, we use two main types of datasets – stories and emails. First, we formulate the problem as an open-ended content generation task, for this purpose we test and validate the systems on a large dataset of stories, called the WritingPrompts [27] dataset collected from Reddit. A more specific hypothesis consists of generating email textual content – for this purpose

we use two organizational email corpora – Enron [78] and Avocado [102]. The following sections give an overview of each dataset along with highlighting some of their statistical properties.

### 3.2.1 WritingPrompts Dataset

The **WritingPrompts dataset** [27, 79] consists of 303,358 pairs of prompts and manually written stories. The dataset was collected by scraping three years of prompts and associated stories from Reddit's WritingPrompts forum.[9] The dataset was built through crowdsourcing on an online community called WritingPrompts where users can submit premises to stories or prompts and can invite submissions from other online users. Each prompt or premise can have multiple story submissions, varying in length, topic and structure. The submitted story responses should follow or be inspired in some manner by the prompt. For other details about this dataset, we refer the readers to [27]. In this dissertation, we aim to generate open-ended textual content from given set of conditioning prompts. Therefore, the WritingPrompts dataset was chosen since it provides a collection of conditioning prompts which act as a guide to the machine for generation and human-written stories which act as baselines for comparison with the machine-generated text. Table 1 shows an example of a story-prompt pair from the WritingPrompts dataset.

Table 1: An example Prompt-Story pair from the WritingPrompts dataset

| | |
|---|---|
| **Prompt** | Pick a singer and write a song/short story from only the words in their songs. |
| **Story** | Single song word pool : "upward over the mountain"- iron & wine, I had an eye for the girl that night, mother she sold her body with some friends on the corner and I had money I saved for the weekend. she got in the car and I had taken her to where I lived upward over the mountain. I gave her my coat when we were in the garden. [...] |

For evaluation, the dataset was split into three parts: training (90%), testing (5%) and validation (5%). Following the preprocessing steps in [27], the stories from the dataset are truncated to the first 1000 words for the experiments. For model fine-tuning, we preprocess the original dataset to create

[9] `www.reddit.com/r/WritingPrompts/`

two new datasets using the Byte Pair Encoding (BPE) tokenization scheme – WritingPrompts-512 and WritingPrompts-1024, where the cut-off BPE length is 512 and 1024 tokens respectively. The detailed data preprocessing steps along with the dataset statistics are given below.

**Dataset Preprocessing.** We describe the preprocessing steps followed to prepare the datasets for model fine-tuning, training and evaluation. The authors of [27] have provided a fairly clean and pre-processed version of the WritingPrompts dataset.[10] Table 2 summarizes the statistics of the truncated WritingPrompts dataset used in experiments.

Table 2: Statistics of truncated WritingPrompts dataset. Here, $S$: Story, $P$: Prompt, $|V|$: Vocabulary, $WC$: Word count, L_avg: Average length in words

| | $\#\text{pairs}_{S-P}$ | $\lvert V\rvert_S$ | $\lvert V\rvert_P$ | $WC_S$ | $WC_P$ | $\overline{L_S}$ | $\overline{L_P}$ |
|---|---|---|---|---|---|---|---|
| **Train** | 272,600 | 454,219 | 39,892 | 160M | 7.7M | 590 | 28.4 |
| **Test** | 15,138 | 83,862 | 11,162 | 8.9M | 425K | 592 | 28.1 |
| **Valid** | 15,620 | 85,576 | 11,129 | 9.1M | 454K | 585 | 29.0 |

For the purpose of fine-tuning the language models, we create three additional datasets - WritingPrompts-256, WritingPrompts-512 and WritingPrompts-1024. These datasets are more suitable for the limited context size[11] of the GPT2 models that will be fine-tuned on the prompt-story dataset. The BPE tokenization scheme used to tokenize the $(prompt, story)$ pairs selects the instances having a total token length equal to a certain given threshold (BPE token length). Thus, for this research, the BPE token lengths chosen were 256, 512, and 1024 respectively. The tokenization was done using the BPE model for English provided by the Python library 'BPEmb.' [12] Table 3 summarizes the statistics of the resulting preprocessed datasets. Note that the WritingPrompts-1024 dataset is a better representative of the original dataset than WritingPrompts-512 and WritingPrompts-256, as demonstrated by the descriptive statistics. For the model fine-tuning, we use instances from the WritingPrompts-512 and -1024 datasets.

[10] `https://dl.fbaipublicfiles.com/fairseq/data/writingPrompts.tar.gz`
[11] the maximum BPE token size (here, 1024) that the model can process
[12] `https://nlp.h-its.org/bpemb/\#usage`

Table 3: Statistics of truncated WritingPrompts (wP)-256, -512 and -1024 datasets.

| **wP**-BPE | Type | #pairs$_{S-P}$ | $\|V\|_S$ | $\|V\|_P$ | $WC_S$ | $WC_P$ | $\overline{L_S}$ | $\overline{L_P}$ |
|---|---|---|---|---|---|---|---|---|
| **256** | **Train** | 12,430 | 41,140 | 11,911 | 1.9M | 239K | 155.3 | 19.2 |
| | **Test** | 710 | 9,988 | 2,413 | 110K | 14K | 155.8 | 19.4 |
| | **Valid** | 738 | 10,416 | 2,279 | 115K | 13K | 156.1 | 18.1 |
| **512** | **Train** | 71,450 | 123,930 | 26,087 | 18M | 1.6M | 256.9 | 22.6 |
| | **Test** | 3,961 | 30,874 | 6,542 | 1M | 90K | 257.8 | 22.7 |
| | **Valid** | 4,169 | 31,081 | 6,572 | 1.1M | 95K | 257.2 | 22.8 |
| **1024** | **Train** | 171,844 | 268,390 | 35,009 | 74.2M | 4.2M | 481.7 | 24.5 |
| | **Test** | 9,536 | 57,392 | 9,546 | 4.1M | 232K | 434.2 | 24.3 |
| | **Valid** | 9,978 | 58,127 | 9,524 | 4.3M | 249K | 430.9 | 24.9 |

### 3.2.2 Email Corpora

We use multiple collections of email datasets for training and evaluation of the generative language models and architectures that we propose in this research. To automate text generation emulating the writing style of an individual, it is necessary to build a generative language model on a corpus of human-written legitimate emails. Moreover, most current phishing attacks are targeted at corporations.[13] For training the email generation models, we use two large sources of human written, legitimate, organizational emails – the publicly available Enron Corpus [26] and the Avocado Corpus [23]. The other legitimate sources used for training are from Wikileaks (Sarah Palin and Hilary Clinton emails) and phishing sources include the datasets used in [21] for training.

The **Enron** corpus [26] is a large dataset of 500,000+ emails from a Houston-based company, Enron.[14] The original corpus consisted of emails sent and/or received by 158 employees (mostly senior management of Enron) between the years 1998 and 2004 including advertisements, spam and other mails. The data was posted online in 2004. The dataset used is a sanitized (removal of full headers, etc.) version of the original corpus and became public in 2015. The authors in [78] used a clean version of the Enron corpus with 200,000+ human-written emails that were extracted from the sent and/or received folders of the Enron employees – we assume these human authored emails and email threads to be *legitimate* for the scope of this study.

[13] https://www.ic3.gov/media/2019/190910.aspx
[14] http://www.ahschulz.de/enron-email-data/

The **Avocado** corpus [23] is a collection of 900,000+ emails from a defunct information technology company, here referred to as "AvocadoIT" provided by the Linguistic Data Consortium (LDC).[15] The corpus contains the full content of emails, with anonymized named entities and additional meta information, extracted from Outlook mailboxes for 279 company employees sent between the years 1995 and 2003. Extracting emails of size[16] 20 or more, from the email threads for the analysis, we get 530,008 email messages. Due to academic licensing restrictions, the Avocado data has been relatively less used. Many researchers view the Avocado corpus as richer and better than the Enron corpus – primarily because access to Avocado is controlled by LDC and it is strictly for research. Avocado can only be accessed with the consent of the legal owner of the corpus while Enron was made public with no agreement required for research, thus making it accessible to researchers and attackers.

Apart from the legitimate sources mentioned above, we also consider **Wikileaks** dataset – 48 emails sent by Sarah Palin and 55 from Hillary Clinton obtained from the archives released in [93, 103] respectively. For modeling **phishing or malicious** elements into the generated content, we also experiment with **phishing** datasets. We used two malicious sources of data – (a) 197 phishing emails collected by the authors of [21] and (b) 3392 Phishing emails from Jose Nazario's Phishing corpus.[17]

Below, we summarize the preprocessing steps along with the statistics of the datasets. We summarize the properties of the Enron and Avocado email corpora in Table 4 and the same for the other email datasets in Table 5.

**Data Preprocessing.** Both these datasets are large and sparse in both topic and information. For this study we apply pre-processing techniques to select emails of length lesser than or equal to a given Byte Pair Encoding or BPE token size threshold i.e., the maximum number of BPE tokens that appear in the text. For fine-tuning the Avocado and Enron datasets, we use the Byte Pair Encoding (BPE)-based tokenization scheme used in [81] to limit the size of the instances in

[15] `http://ic4f.me/projects/avocado`
[16] number of words
[17] `http://monkey.org/~jose/wiki/doku.php` (2004), Deprecated now

the training dataset. The tokenization of the data ensures the balance between character-level and word-level hybrid representations of the textual content. The BPE tokenization threshold sizes we use are 128 and 512 respectively. Table 4 lists descriptive statistics of the tokenized and original email corpora. We train and evaluate the systems on the training (90%), validation (5%) and testing (5%) partitions of the tokenized email corpora. For this research, we provide the evaluations on the Enron-512 and Avocado-512 datasets.

Table 4: Statistics of ENRON and AVOCADO datasets (Original and BPE-tokenized). Here, orig: Original, $|V|$: Vocabulary, $WC$: Word count, $S$: Total number of sentences, $\overline{S}$: Average number of sentences, $\overline{SL}$: Average Sentence length in number of words

| | **Size** | $WC_{total}$ | $\overline{WC}$ | $\|V\|$ | $\overline{SL}$ | $S_{total}$ | $\overline{S}$ |
|---|---|---|---|---|---|---|---|
| **Enron-orig** | 252,759 | 81M | 321 | 1.7M | 32 | 2.4M | 9 |
| **Enron-128** | 52,450 | 2.1M | 40 | 111K | 18 | 120K | 2 |
| **Enron-512** | 114,147 | 13M | 116 | 420K | 25 | 494K | 5 |
| **Avocado-orig** | 530,008 | 93M | 243 | 2.9M | 18 | 11M | 7 |
| **Avocado-128** | 211,452 | 7.4M | 35 | 100K | 14 | 53K | 3 |
| **Avocado-512** | 340,453 | 26M | 78 | 260K | 18 | 139K | 4 |

Table 5: Additional email corpora statistics. $|V|$: Vocabulary, $WC$: Word count

| | **Size** | $\overline{WC}$ | $\overline{\|V\|}$ |
|---|---|---|---|
| **Clinton** | 48 | 32 | 21 |
| **Palin** | 55 | 33 | 26 |
| **Verma Phish** | 197 | 153 | 99 |
| **Nazario Phish** | 3392 | 210 | 129 |
| **Verma Phish-Fil** | 127 | 50 | 34 |
| **Nazario Phish-Fil** | 2148 | 115 | 71 |

On close inspection, phishing emails suffer from presence of incoherent HTML content which can pollute the training model. Therefore, from the original data, we carefully filter out the non-English emails and emails with full HTML content. This step, however, was limited to only phishing emails. The filtered datasets are shown in Table 5. We also enlist the other data preprocessing schemes we use to clean the email datasets. These phishing emails have certain textual cues which have been studied and identified in previous research. We assimilate and mention those features in Section 3.6.

- Removal of special characters like @, #, $, % as well as common punctuations from the email body along with removal of all trailing spaces, newline characters, etc., from the text body.

- Replacement of all named entities (person, location, etc.) with *ent* tag along with replacement of URL links with the *link* tag and email addresses with $emailID$ tag. Emails usually have other URLs or email IDs. These can *pollute* and confuse the learning model as to what are the more important words in the text. Hence, after the normalization step, the replacement of named entities with the <NET> tag. We use Python NLTK NER[18] for identification of the named entities.

- Adding the *End of Text* (| <endoftext> |) tags after the email bodies to respectively mark the beginning and end of the content while training and/or evaluating.

- Sanitization of non-ASCII characters from the text and removal of HTML text fragments and broken links from text body.

## 3.3 Deep Neural Architectures and Models

We first explain some popular baseline deep neural architectures used to build a generative language model. These are usually RNNs (specially LSTMs) and Seq2Seq and Transformer networks. We then briefly describe the existing pre-trained language models and finally discuss some existing hierarchical or hybrid architectures.

### 3.3.1 Recurrent Neural Network Architecture

Traditional pre-neural language models like N-grams are limited by the history or the sequence of the textual content that these models can look back upon while training. However, RNNs can retain the information provided by some text sequence, making it work as a "memory"-based model. While building a model, RNNs cannot preserve long term dependencies – thus making Long Short Term

[18] https://www.nltk.org/book/ch07.html

Memory architectures (LSTM) networks more popular. LSTMs can learn a better language/text representation than RNNs for longer sequences of text [88].

Textual content can be considered as a sequence of words and characters placed together to convey meaningful information. In text generation, deep neural architectures have seen unprecedented success in emulating one's writing when trained on huge amounts of written textual content [108], [34], [90]. Recurrent Neural Networks (RNNs) can retain information learned from text sequences – helpful for learning representations of word sequences in the input text. The trained language model can subsequently generate samples similar in form and context to the input data. The architectures use *words* as input units for generation [106], [45] by leveraging LSTMs as building blocks for learning the language model.

We leverage this ability of RNNs for a proactive study - generation of targeted emails suitable for spear-phishing or masquerading attacks. We use such a model in generating textual content (emails) automatically in the evaluation in Section 4. Figure 3 shows the overall model for word prediction using Bi-LSTMs as the baseline unit.

The word-based RNN is trained by feeding the LSTM network a sequence of word units from the training dataset sequentially. Here, we use Bi-LSTM or Bidirectional LSTM networks that are twice as efficient as regular LSTMs, with one set of networks learning forward sequences and the other set learning backward sequences [18]. While training, the entire training dataset is divided into sequences of one-hot vectors of words, each sequence has a specific length chosen based on the nature of the data. These sequences of training data are fed into the RNN architecture (Bi-LSTM) to learn a language model to predict a sequence of words based on a step size. The model training parameters include the number of epochs, the batch size in which the data is fed. The model architecture has the features – RNN size or number of RNN nodes, the dropout rate, the learning rate and the optimization function. We use Python's Keras library to design and implement the word-based RNN architecture. The usual sequence step size is one word unit and the sequence size for textual content is 20. When the model is generating a sample, the input to the text generation system is a one-hot vector sequence of words ($w_0$, $w_1$, $w_2$, ..., $w_{N-1}$) where $N$ is the sequence

length. At every time step $t$ (starting with 0), we feed this sequence of words, the output in $N+1$-th predicted by the hidden layer. Through error optimization,[19] the model learns to capture the best representation of coherent word sequences that constitute the textual content to be generated.

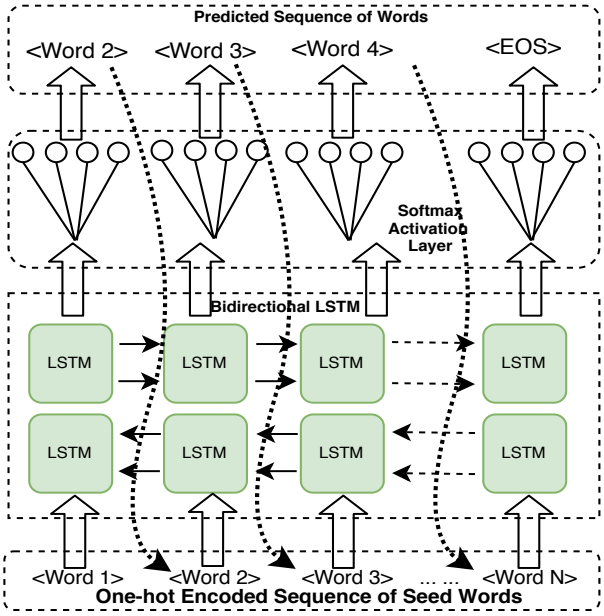



Figure 3: Word generation model with Bi-LSTMs

### 3.3.2 Transformer-based Language Models

We provide an overview of the pre-trained and the fine-tuned massive transformer-based language models that we use in this study.

**Pre-trained Language Models.** We review the publicly available large-scale **pre-trained language models** provided by OpenAI and Google. These models have yielded exemplary results on a variety of natural language understanding based applications even when applied in a 'zero-shot' setting [79, 95]. Recently the following language models are gaining popularity in text generation tasks [66, 2, 48]. OpenAI's **GPT** [72] model has 110M (**openai-gpt**) parameters, and the subsequently released three **GPT2** models [73] have respectively 117M (**gpt2**), 345M (**gpt2-medium**) and 774M (**gpt2-large**) parameters. Apart from the GPT models, there are **XLNet** [107] and

[19] the optimization function, e.g., *Adam* [55]

**Transformer-XL** [19] models from Google-CMU, which have been shown to generate comparatively better textual content than GPT2 models [2]. There are two pre-trained XLNet models – **xlnet-base-cased** and **xlnet-large-cased** with 110M and 340M parameters respectively, and the Transformer-XL model (**transfo-xl-wt103**) with 257M parameters. We refer the readers to the cited papers to explore these architectures in detail. The Fusion Model proposed by [27] is a Seq2Seq architecture and has a vocabulary of 104,960 words, a 3-layer encoder and 8-layer decoder in the first seq2seq model, and a 5-layer encoder and 5-layer decoder in the second model – in total, 255.4 million parameters.

**Fine-tuned Language Models.** While experimenting with pre-trained language models may be feasible for demonstrating a proof-of-the-concept application, an in-depth study and evaluation for a specific task would require model retraining or **fine-tuning**. **Fine-tuning** refers to model retraining on a task and domain specific dataset without largely modifying the architecture, to further tune the model to the specific data to be evaluated on. However, fine-tuning such huge transformer models can be computationally expensive. The details of model fine-tuning or retraining have been explained based on the dataset in Section 5.1. The basic modules of the generative GPT-2 architecture based on Transformer decoder-only blocks is shown in Figure 4.

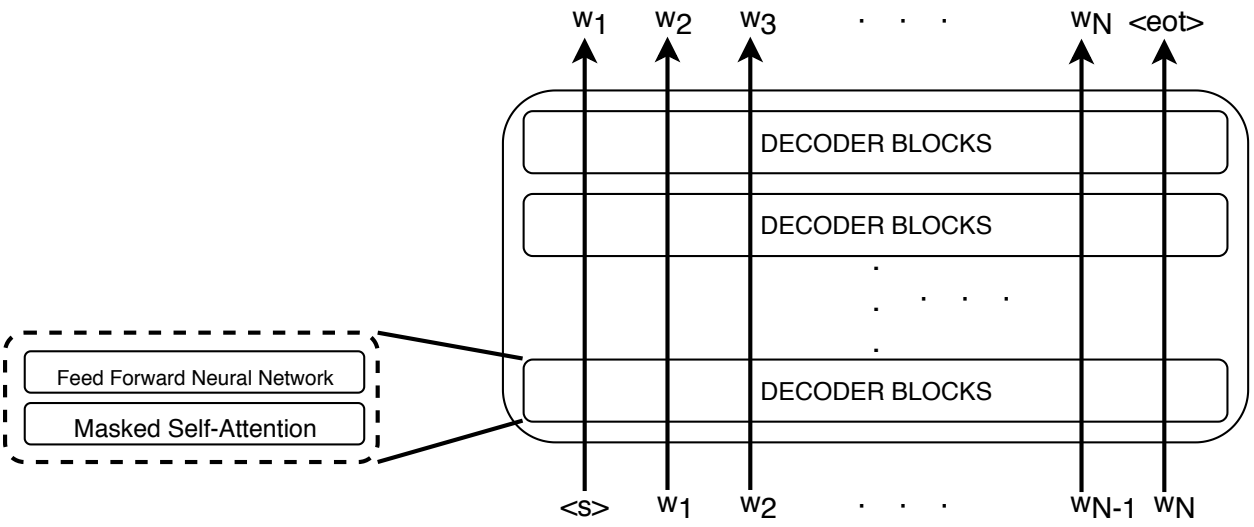



Figure 4: GPT-2 Architecture Schematic

### 3.3.3 Neural Sentence Embeddings

Inspired by word-based embedding methods, the document embeddings can map documents to informative vector representations. A document-based representation teaches the learning algorithm to review the specific meaning of the units (sentences, words, etc.) in the document along with their syntactic role.[20] We use three different types of neural document-based representations which creates the vectors at the sentence level. The representations used are Doc2Vec [57], Sent2Vec [67] and Sentence BERT [74]. We explain the representation methods briefly in this section.

**Doc2Vec.** The document to vector representations learns a unique vector for each document on which the model is trained. A Doc2Vec or D2V vector represents each document as an average of embeddings of words randomly sampled from the document on which the vector is built. It is a paragraph vectorization based technique used to generalize and extend word-based vectors like (word2vec) to word sequences. The Doc2Vec representation method proposed by [57] is similar to the Word2Vec representation model and added a Paragraph ID. Therefore, when training the word vector, the document vector is trained as well, and of the end of training, it holds a numeric representation of the document. We use the Python library Gensim[21] to implement the Doc2Vec vectorization method with the *distributed memory* algorithm. The Doc2Vec implementations are used to create 500-dimensional sentence-level embeddings for the predictive model in the proposed generative architecture.

**Sent2Vec.** These unsupervised representations are an extension of word2vec vectorization techniques. The Sent2Vec model can compose sentence embeddings by using word-level vectors and n-gram embeddings, therefore handling out-of-vocabulary word fragments while learning the embeddings for the sentences. In such representations, the whole sentence is considered the context and the possible class labels are vocabulary words. The Sent2Vec embedding scheme was proposed by [67]. We use a Python-based implementation of Sent2Vec for evaluation purposes.[22] This

[20] https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d#e3d4
[21] https://radimrehurek.com/gensim/models/doc2vec.html
[22] https://github.com/epfml/sent2vec

implementation was used as an evaluation metric for measuring inter-sentence connectedness (Section 3.5). The text was encoded using the pre-trained bigram model trained on English Wikipedia articles to create 700-dimensional vectors.[23]

**Sentence BERT.** BERT (or Bidirectional Encoder Representations from Transformers)-based models have already emerged as the state-of-the-art techniques in many language understanding based tasks [22]. Using fine-tuned BERT-based models to extract sentence level representations from documents has largely improved regression-based tasks as shown in [74]. The authors propose Sentence BERT or S-BERT – the technique uses BERT-based neural pretrained models to generate dense vector representations for sentences and paragraphs from the given document. The implementation of S-BERT used here is the *bert-base-nli-stsb-mean-tokens* pre-trained language model which uses BERT-base-uncased as the base model trained on NLI and STSb benchmark datasets.[24] We use S-BERT to create 768-dimensional sentence-level embeddings for the predictive model in the proposed generative architecture.

## 3.4 Decoding Techniques and Sample Generation

A decoding algorithm is used to generate the text from the trained or fine-tuned language model. These are essentially sampling techniques, used to select words or character units from the distribution built by the language model. Among the popular techniques used, two prevalent algorithms used before are **greedy decoding** [37] and **sampling-based decoding** [21]. We also briefly discuss the **temperature($\tau$)-based** sample generation technique to control the diversity of the generated content during decoding step. The newer methods for sample generation include **top-$k$ sampling and nuclei-based sampling** [72].

### 3.4.1 Greedy Decoding

A simple algorithm wherein at each step, the most probable word is selected from the trained language model built on the input dataset. This generated word is then used as the next word and

[23] https://drive.google.com/open?id=0B6VhzidiLvjSaER5YkJUdWdPWU0
[24] https://www.tensorflow.org/datasets/catalog/glue

fed as input to the generative language model on the next step. This is repeated until one reaches the <END> tag (i.e., the stop token) or reaches a preset or desired maximum length. Below is the given simple decoding algorithm in Equation 1. The $W_t^j$ is a generated unit $j$ at time instance $t$.

$$P(W_t^j) = argmax(P(W_t^j|W_0^j W_1^j ... W_{t-1}^j)) \tag{1}$$

However, due to the greedy selection scheme, output can be poor (e.g., ungrammatical, unnatural, nonsensical). Figure 5 gives an idea on how the *argmax* decoding algorithm works. The algorithm selects the word with the highest probability at each step and feeds it back into the next step.

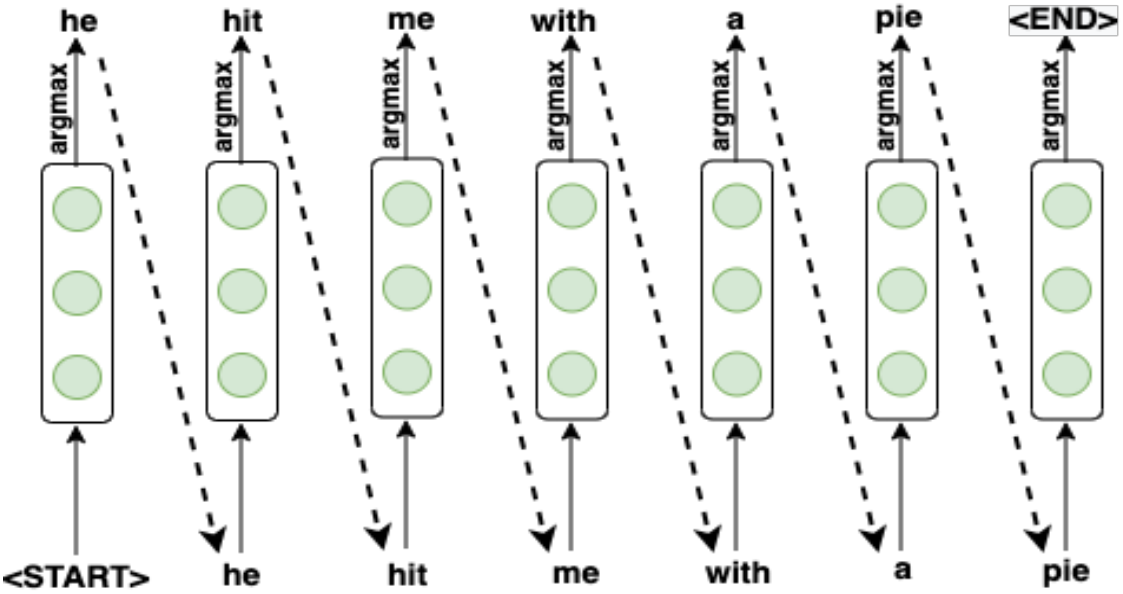



Figure 5: Greedy Sampling Technique

### 3.4.2 Sampling-based Decoding

This consists of random sampling and top-$k$ sampling used to select the next generated word. In **random sampling**, in each step $t$, the next word is randomly sampled from the probability distribution modeled on the source documents. The approach is similar to greedy decoding, but we sample the output word randomly instead of *argmax*. For sample generation, the authors

in [27, 73] use **top-$k$ random sampling** scheme. The algorithm is neither as greedy as a likelihood maximizing algorithm, nor as non-deterministic as unbiased sampling. This method trims the probability distribution over the vocabulary to keep the top $k$ tokens at each timestep. The values of $k$ are changed to get newer generated samples. The papers [73, 27, 79] discuss the efficacy of this technique over conventional beam search and greedy methods [32] and experiment with different values of $k$ for sample generation. Both these sampling-based techniques empirically prove to be more efficient than prevalent decoding methods like greedy sampling [27, 72].

### 3.4.3 Temperature-based Sampling

During the generation phase, we feed a sequence ($N$) of seed words ($W_{seed_0}, W_{seed_1}, W_{seed_2}, ..., W_{seed_N}$) into the trained model, used to start off the word generation system. When the model gets a seed word ($W_{seed_0}$) as input, it outputs the next word ($W_1$) by selecting the one most likely to occur after $W_{seed_0}$ depending on the conditional probability distribution, $P(W_1|W_{seed_0})$. When this step is extended for an input sequence of $N$ seeds, the model (Figure 5) can generate a text body of $N+1$ words, the $N+1^{th}$ word being the output.

The final layer of a generative neural language model (Section 3.3), which calculates the above conditional probability, is a $softmax$ normalization which is used for computing the distribution for the next word followed by subsequent sampling. We use $temperature(\tau)$ as the hyper-parameter for selecting the word samples - regulating the parameter $\tau$ in Equation 2 encourages or controls the diversity of the generated text. The novelty or eccentricity of the generative model can be evaluated by varying the *temperature* parameter between $0.0 < \tau \leq 1.0$. While lower values of $\tau$ generate relatively deterministic samples, higher values can make the process more stochastic.

Equation 2 shows the probability distribution built by the model for the sequences of words along with the incorporation of *temperature* control parameter($\tau$), $P(W_{t+1}|W_{t' \leq t}) = softmax(W_t)$. Here, each $W_t$ is an $n$-dimensional vector where $n$ is the size of the word vocabulary and $w_t^j$ represents the component of the output corresponding to the word $j$ in the vocabulary, at $t$-th time

step, for a given $\tau$.

$$P(softmax(W_t^j)) = \frac{e^{\frac{w_t^j}{\tau}}}{\sum_{k=1}^{n} e^{\frac{w_t^k}{\tau}}} \tag{2}$$

### 3.5 Evaluation of Generated Content

We divide the set of metrics into five major groups based on their domain of evaluation – readability, syntactic style and complexity, part-of-speech usage, measure of coherence and prompt-based conditioning.

#### 3.5.1 Prompt-based Conditioning

The models must be able to condition well on the given story prompt, which means that the text generated must relate to a given initiating premise. Conventional models often fail to produce text that is semantically and contextually related to a given prompt/seed [27]. A provided sample seed acts as a guide to the language model to start generating content based on a specific theme. For example, if one were to write a story, a given prompt acts as a guide to generate the content specific to a particular topic. Similarly, for email generation, the email subject may act as a good seed to start the generation of content pertaining to a particular intent. In such a case, measuring overlap with the seed gives a good indicator of the level of semantic relatedness between the generated content and the starting prompt. A higher score of overlap (close to 100%) can also be due to samples containing only words from the prompt repeatedly. So the targeted level of overlap percentage should be close to human written content – since, hypothetically speaking humans tend to reuse words from the given prompt to a lesser extent.

For the implementation, using the Python NLTK toolkit [61], we investigate the percentage overlap [79] of uni-, bi-, and tri-grams between the generated stories and the prompt. As a preprocessing step, we use NLTK's word tokenizer with stopword elimination of common English words. During the metric evaluation in Sections 4.2, 5.2 and 6.4, we show that there exists a high positive correlation among **n-grams** ($n \in \{1, 2, 3\}$). Moreover, the percentage values of the bigram overlap are neither as high as unigram nor as low as trigram overlap percentage. Therefore, we choose to

report bigram overlap percentage of generated instances with the conditioning prompt.

### 3.5.2 Style and Complexity

While there is no good way to measure stylistic complexity of textual content, an overly complex piece of text can reduce readability while poorly written content demonstrates lack of sophistication [79]. Along with considering the mean (L_avg) and standard deviation (L_sd) of the sentence length in the stories, we study the type token ratio (as percentage, *ttr_pc*) to observe how the stylistic quality of the generated text compares to human writing.

**Sentence Length.** This property has been used in previous research to estimate the level of syntactic complexity [54, 79, 77]. It has been considered a reliable metric that can capture text genre and overall content readability by previous researchers. We use Python's NLTK library to count the number of word tokens in the generated content. We study the average ($\mu_{SL}$ or $L_{avg}$) and the standard deviation (S.D.) ($\sigma_{SL}$ or $L_{SD}$) of sentence length in the generated textual content. The statistical properties of the number of word tokens in the generated sentences show if the models usually generate longer sentences or shorter sentences, how the sentence length varies through the generated content body. This helps to correlate the model performance with the human writing style.

**Type Token Ratio.** Type-token ratio (TTR) [77] measures complexity, lexical richness, or variety in vocabulary. This metric sheds light on the following – is there an occurrence of the same words over and over in the generated text? Is there a variety of different words generated by the language model while conditioning on a prompt? TTR is the ratio between the total vocabulary[25] or *types*, and the total number of words or *tokens* in a given piece of textual content. A higher TTR value indicates greater lexical richness of the text. The metric helps to measure the lexical richness of the generated content and compare the same with human writing.

[25] number of unique words

### 3.5.3 Readability Measures

Readability metrics are an automatic and easy measurement of text difficulty [65, 52]. Readability scores like Flesch Reading Ease (**fre**) and Flesch-Kincaid Grade Score (**fkg**) [54] and Dale-Chall Readability score (**dcr**) [69] attempt to quantify the level of difficulty of a text with respect to the reader's education level. While in FRE and FKG, text complexity is calculated using the average length of the sentence as well as presence of polysyllabic words; the DCR score takes into account familiarity or knowledge of a word while calculating readability. We use Python's *textstat* [9] library to calculate the different readability metrics for the generated content and compare it with the human authored text.

### 3.5.4 Measure of Coherence

It is important to evaluate the coherence in written content to determine whether there is correlation across the units (sentence or words). We propose the sentence connectedness (**sent_conn**) metric to evaluate the cohesion at the sentence level in textual content. Using Sent2Vec-based embeddings, we transform each sentence in the story to their embedding vectors. To capture the difference in sentence meaning, we calculate the angular variation in consecutive sentence vectors in radians. We compute the standard deviation of the list of pairwise angular differences obtained for each adjacent sentence pair. The lower the variation, the greater is the *connectedness* or coherence across the text content.

This metric is for cohesion at the sentence level. Using Sent2Vec-based embeddings and a pre-trained Wikipedia-based Bigram model,[26] each sentence token is vectorized. The angular variation between sentence vectors are calculated in radians – we calculate the mean ($\mu$) and standard deviation ($\sigma$) of the total set of variations. Finally, the harmonic mean (H.M.) of the mean and standard deviation is calculated (**HM_sent_conn**). Lower values of harmonic mean would indicate less angular variation and deviation between vectors. The Sent2Vec implementation used has been discussed in detail in Section 3.3.3. The embeddings are generated using the pre-trained

[26] https://github.com/epfml/sent2vec

language bigram-based model - *sent2vec_wiki_bigrams*, a 16GB sized language model consisting of 700 dimensional vectors trained on English articles from Wikipedia.[27]

### 3.5.5 Part-of-Speech Usage

Part-of-Speech (POS) usage can be a useful indicator of linguistic quality. The distribution of parts-of-speech (POS) tags in textual content provides information like similarity in authorship, rarity of word usage, and originality of POS tags. An exploratory analysis of the textual content in existing research reveals that **Noun** and **Verb** tags are the most commonly occurring parts of speech [77]. Hence, we primarily compare the frequency distributions of these two tags between the synthetic and human written references.

Here, we report the percentage of NOUN and VERB tags that appear in the generated text with human stories acting as the baseline. The Spacy POS tagger[28] for Python was used for tagging purposes.

## 3.6 Textual Features in Malign Emails

This section presents a list of cues usually observed in spoofing emails that demarcate such attack vectors from their legitimate counterparts. Highlighting and studying such common signals helped us prepare, process and evaluate the training data as well as the generated emails.

Use of textual features, like presence of common action words, organization names, poisoned links to malicious webpages of financial institutions, grammatical errors, etc., is common in phishing email detection methods [97], [98], [100]. Moreover, researchers have widely studied spam, phishing, spear phishing emails to identify common signs that appear across malicious emails [24], [30]. However, since such signals are certain signs of malicious intent an attacker would consciously avoid incorporating these words in a targeted email. Assuming this, the proposed generator should also learn to identify and eliminate overuse of such words.

Therefore, we curate a list of textual cues frequently used in spoofing emails after careful review

[27] https://github.com/epfml/sent2vec#generating-features-from-pre-trained-models
[28] `https://spacy.io/usage/linguistic-features`

of phishing email literature [24], [97], [98], [100], [30], [4]. The list of these textual cues along with examples have been provided in Table 6. Researchers prefer to train their proposed detection methods on publicly available data. Since phishing emails are fairly rare, we base this evaluation on the largest publicly available dataset of malicious emails: Nazario Phishing Corpus.[29] The Base-64 encoded HTML content in the emails is filtered out and finally 3,392 fairly clean emails with textual content (>10 words) are used to extract the enlisted spoofing cues.

Table 6: Common Spoofing Cues in Phishing Email bodies

| **Feature Types** | **Examples** |
|---|---|
| **Organization Names** | (a) *Financial* like eBay, PayPal, Bank of America, Western Union<br>(b) *Government* like Internal Revenue services, United Parcel Service<br>(c) *Software* like Dell, Microsoft, Apple |
| **Action Verbs and Urgency Adverbs** | (a) *Action verbs* like click, follow, visit, go, update, apply, submit, confirm, cancel, dispute, enroll, login, answer, reply<br>(b) *Adverbs implying urgency* like today, instantly, straightaway, straight, directly, once, urgently, desperately, immediately, soon, shortly, presently, before, ahead, front |
| **Persuasion Principles** | (a) *Authority* like an email from a bank asking the victim to update password of his online account<br>(b) *Social proof* denoted by Emails from the IT department of the target's institution<br>(c) *Distraction* using emails where a target is tempted to click a link in order to receive a prize<br>(d) *Reciprocation* appealing the victim to respond like resetting a password or paying a bill by clicking a link to a fraudulent website |
| **Misspelled Words** | *Typographical errors* like Paypl, Bnk Amrica, etc. |
| **Presence of Links** | URLs to malicious websites like https://www.maybank2u.com.my, etc. |
| **Other languages** | *Non-English words* like Aviso Importante de BBVA, societe, Transaktionen |

While proposed machine learning systems can detect common cues, these detectors largely depend on historical data. To keep up with advanced reinforcement techniques, phishers may

[29]https://monkey.org/~jose/phishing/

also resort to employing sophisticated techniques for improving their attacks. A sophisticated and targeted attack therefore increases the rate of success. Thus, for social engineering based attacks like email masquerading, spear phishing, or targeted phishing, an attacker may choose to avoid such easily identifiable red flags while generating fake emails. Therefore, in this proactive study, we also refrain from overuse of such spoofing cues in the synthesized emails. This has been done to evade detection by humans and trained phishing and spam email detectors.

## 3.7 Intent Analysis in Organizational Emails

For automatically generating targeted emails, the synthetic email body should contain certain lexical cues and intents for an effective structure [16]. In [78], the authors define email *intent* as the motivation of the sender to send a message to the receiver. In such a scenario, the intent of the sender can be identified by taking into account the context of the message as a whole. Following the work in [78], the authors in [102] perform a large scale context based analysis of intents in email conversations.

Both these works have manually annotated email messages from the Enron and Avocado corpora based on different dimensions. Now, we briefly review the email intents and action categories as well as analyses described in [78] and [102]. We then analyze the manually annotated email messages from Enron and Avocado corpora in terms of occurrence of email intents, inter-annotator correlation, word occurrence and textual nature of human-authored emails. The main purpose of this analysis is to identify the best set of words or textual cues to use as prompts for effective email generation in a particular category.

### 3.7.1 Analyzing Annotated Enron

In [78], the authors select the different 'dimensions' or categories by studying the whole email content, to identify the intent of an email. These categories include attributes like the *intent* of the email sender, the *implicit reason* behind sending the email, etc. Additionally, the authors use a task based classification scheme; the task is the expected action of the recipient after reading the email.

Based on previous research, the authors in [78] identify four major dimensions in email messages – email actions, response expectation, source authority along with implicit reason, and number of tasks. For the scope of this dissertation, we only stick to the primary category of 'email act' dimension defined by [14]. We refer the reader to the paper for descriptions of the email categories (for example, request, propose, commit) based on the action intent conveyed by the sender of the message.

Using 1,143 email messages annotated manually by 5 annotators, Sappelli et al. [78] analyze the intent categories in the Enron email corpora. As for the annotation's reliability, [78] reports a fair agreement between sender and the recipient for the email act with a statistically significant $p$ value for Cohen's $\kappa$. We use the annotated dataset of 1,143 email provided by [78] for our analysis – studying correlations and frequency distributions of the email action categories. We also study how lexical properties like vocabulary richness and Flesch-Kincaid Grade level vary across the categories of emails.

**(A) Frequency Distribution based Analysis.** Figure 6 shows the frequency distribution of email action categories in the annotated Enron corpus. The distribution ratios are averages over all annotations, by all assessors. The Figure 6 shows that the category 'deliver' is the most common, with more than 40% emails in this category. Such messages are usually informative in nature like delivering some information (e.g., "FYI"). The second most common category is request-based emails, e.g., questions asked, or requesting someone to do something.

We also review the correlation across the annotators' agreement on the action category in an email. Figure 7 gives the distribution of the annotated email categories. The heatmap helps us ascertain whether some action categories coexist in an email. For example, some emails may belong to both request and deliver categories. We see that while most categories in the emails from the corpus do not have high correlation, some categories like remind and deliver have a moderate chance of co-existing in the same email. For example, "You need to update your account at the following link. Please see below for the attached link and information." Therefore, an email that has been marked 'deliver' by one annotator, must have been marked 'reminder' by the other annotator.

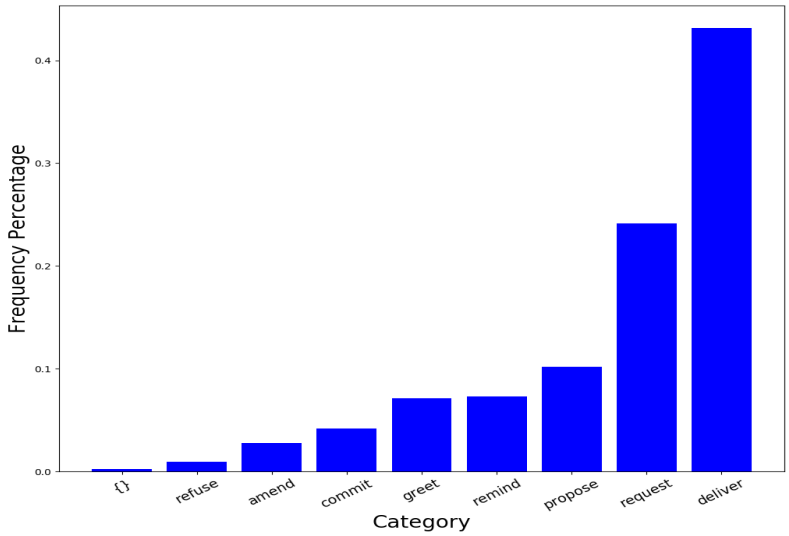



Figure 6: Frequency percentage of email act subcategories for annotated Enron corpus

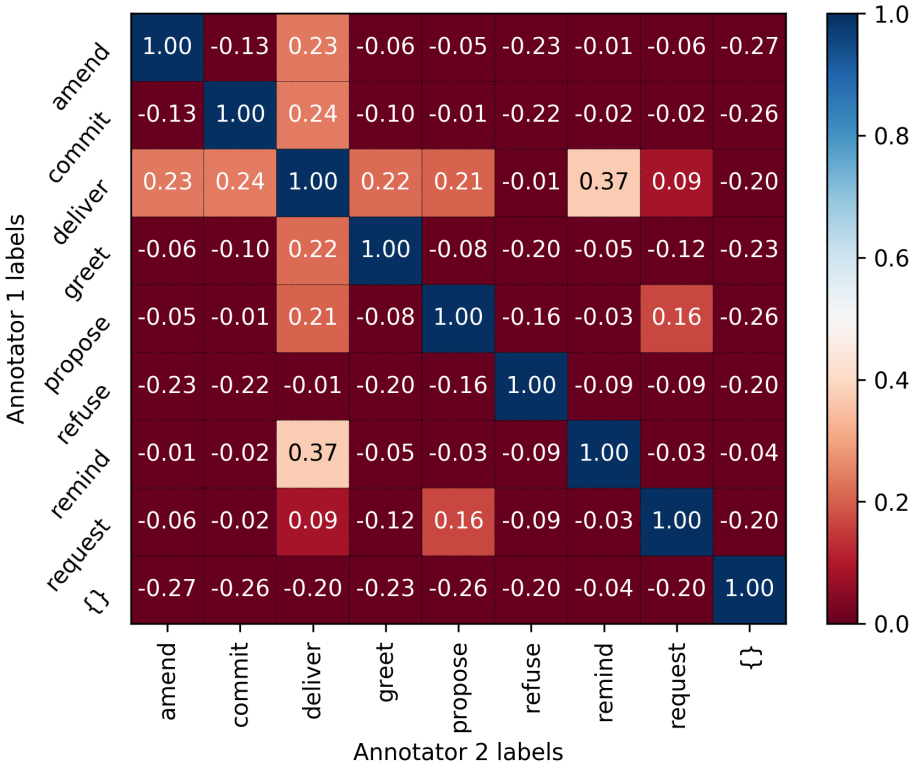



Figure 7: Inter-Annotator Correlation across email act subcategories for annotated Enron corpus

**(B) Lexical feature based Analysis.** We review the lexical attributes across the different email subcategories. Word clouds give an overview of the different high frequency words that appear in a given category of emails. We build word clouds from emails belonging to a selected couple of email action categories. Figure 8 demonstrates the same for the categories *deliver* and *remind* under email act dimension. Some notable observations, we see the words 'meeting' occur in the word clouds for 'remind' category, the word 'deal' and 'time' in the 'delivery' email subcategory.

We use the PhishBench tool proposed in [25] to study how lexical attributes like *Flesch-Kincaid Score* and *Vocabulary Richness*[30] vary in emails across the different categories. The boxplots of these attributes are shown in the subfigures under Figure 9 for FK score and vocabulary richness, respectively. We see that emails in general have limited vocabulary and moderate readability scores.

### 3.7.2 Analyzing Annotated Avocado

Due to a lack of annotated emails from the Avocado corpus, we analyze a set of 1,050 emails from the Avocado corpus. To draw a direct comparison with [102], we provide the annotators with the following email intent categories as defined in the paper: information exchange or IE (sender's intent to *share information* or *request information*), Task Management or TM (sender's intent to *request an action* or *promise an action* pertaining to a task), Scheduling and Planning or SP (sender's intent to *schedule a meeting* or *send a reminder* for an event through email), Social Communication or SC (casual messages like *greetings* or *thank you* notes).

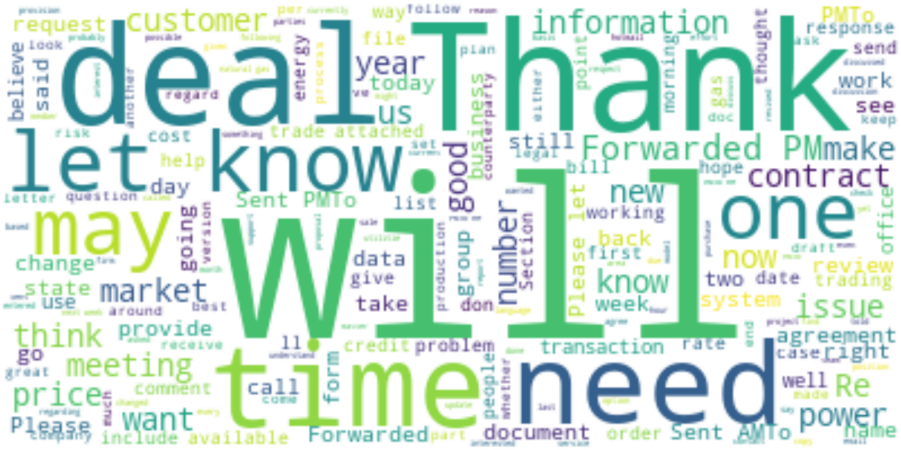
We assign two annotators to annotate each email to the best sub-intent category based on the email message body. If the email is a thread, the annotators are asked to assign the intent based on the first message in the thread. The inter-annotator agreement calculated using the Cohen's $\kappa = 0.475$, shows a *moderate* agreement amongst the annotators.[31]

We now review results of frequency and correlation analysis of the annotated corpus and give an overview of the distribution of lexical features in two categories of emails.

**(A) Frequency Distribution based Analysis.** Figure 10 shows the frequency distribution

[30] Number of unique words in the text or vocabulary

[31] https://www.statisticshowto.datasciencecentral.com/cohens-kappa-statistic/




(a)

(b)

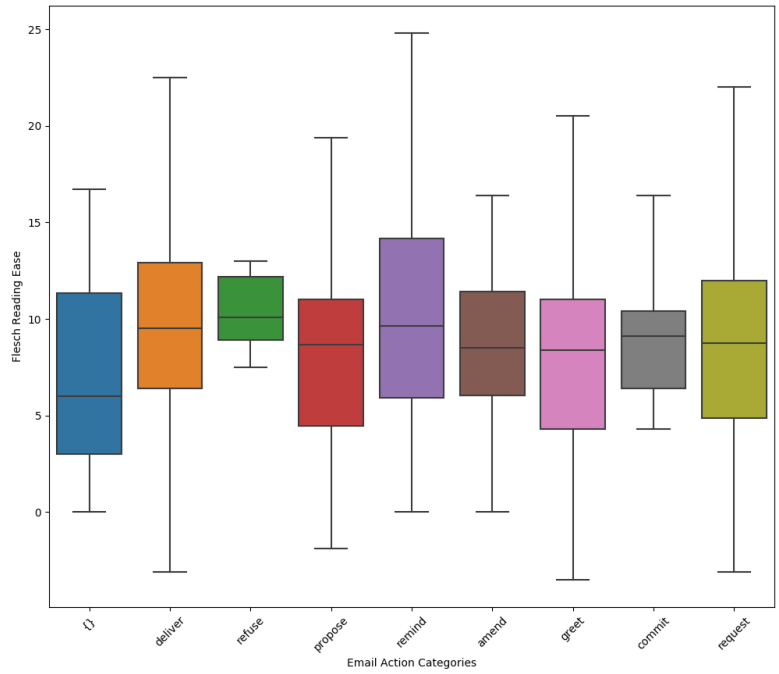
Figure 8: Word cloud for Enron emails under (a) *delivery*. (b) *remind* subcategory

of email sub-intent categories in the annotated Avocado corpus. We calculate the percentages by averaging the number of emails annotated by each annotator for each category. We see that the annotators marked the information sharing category as the most popular with the emails requesting information as the second most. We also review the correlation across the annotators' agreement on the sub-category of emails' intent. Figure 11 gives the correlation of the annotated email categories. It is generated in the same manner as the Enron heatmap for inter-annotator correlation. The heatmap shows the co-occurrence of different categories of intent that may exist within the same

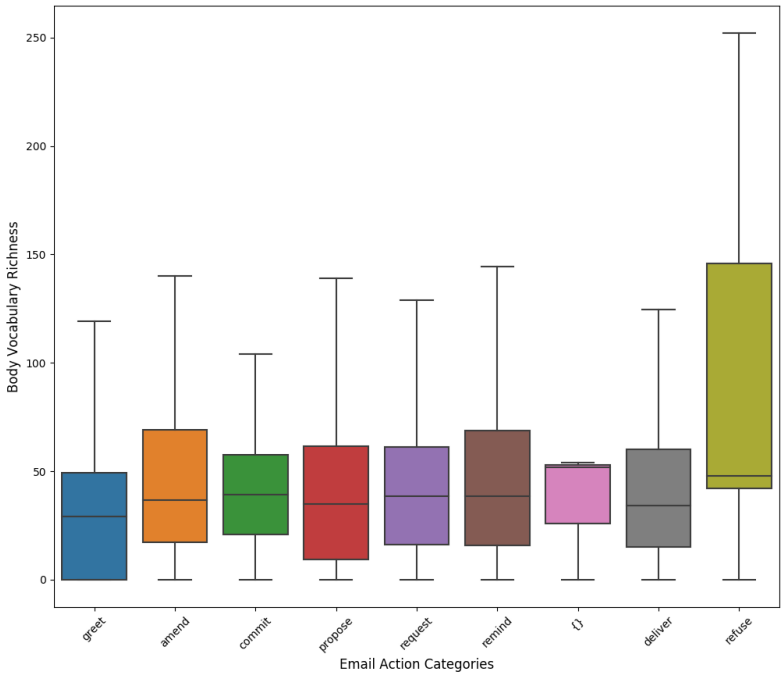
email. For example, sharing of information (IE-ShareInfo) is moderately correlated with almost all the other categories, specially requesting information (IE-ReqInfo), requesting and promising some action (TM-ReqAct and TM-PromAct respectively). This shows that emails generally pertain to more than one specific action or intent.

**(B) Lexical feature based Analysis.** We explore the lexical attributes across the different email subcategories. Word clouds give an overview of the different high frequency words that appear in a given category of emails. We show word clouds from emails belonging to a selected couple of email action categories. Figure 13 shows the email actions *Information Exchange/Share Information* and *Task Management/Requesting Action* respectively. Words like application, customer, meeting, problem, etc., appear in the IE-ShareInfo category while sent, will, need, know, call are some action words in the TM-ReqAct emails.

The subfigures under Figure 12 show the boxplots for the Flesch-Kincaid score and vocabulary richness across email categories calculated using PhishBench [25]. We note that, while most of the categories have statistically insignificant differences in the mean FK score and vocabulary richness measures, the emails for requesting information score much lower. Analyzing the frequency distribution as well as the lexical nature of the annotated emails helps lay the ground for systematically generating the emails using the proposed email generation architecture. Next, we describe the proposed deep neural architecture for generating targeted emails based on intent.




(a)




(b)

Figure 9: Boxplot for Enron emails for (a) FK score. (b) Vocabulary Richness.

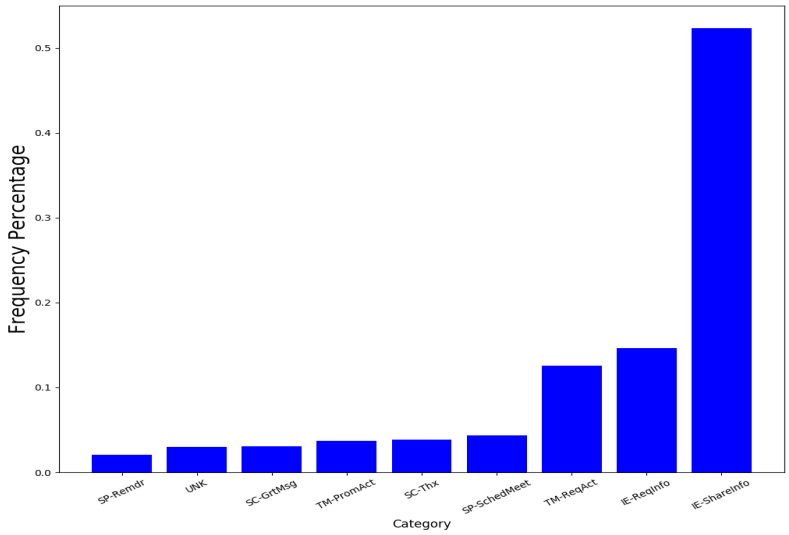



Figure 10: Frequency percentage of email act subcategories for annotated Avocado corpus

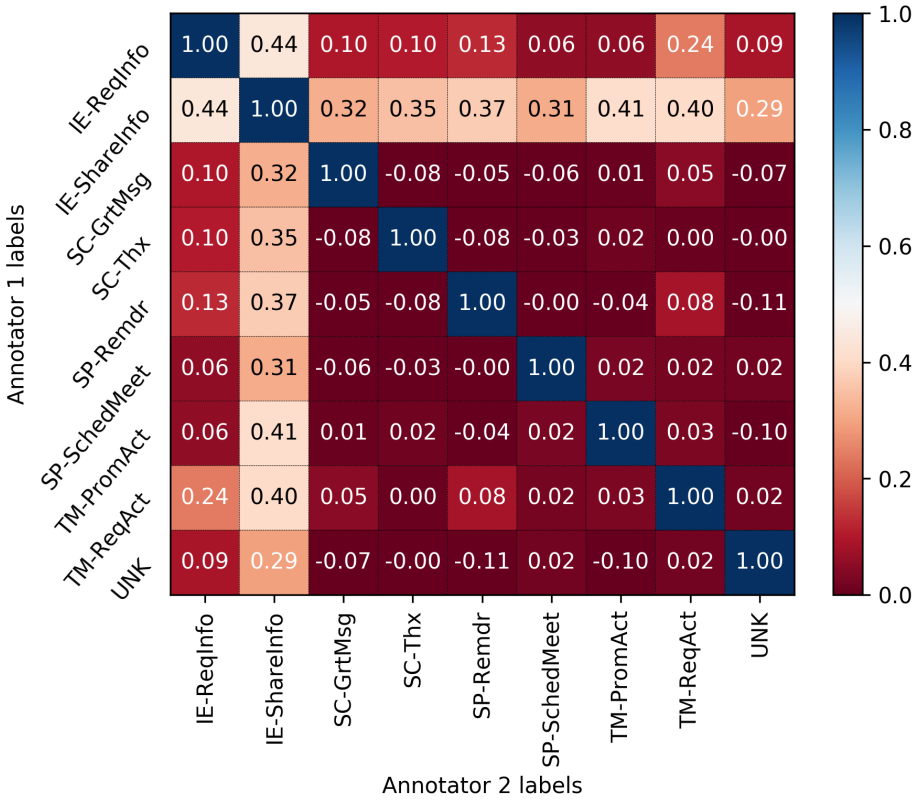
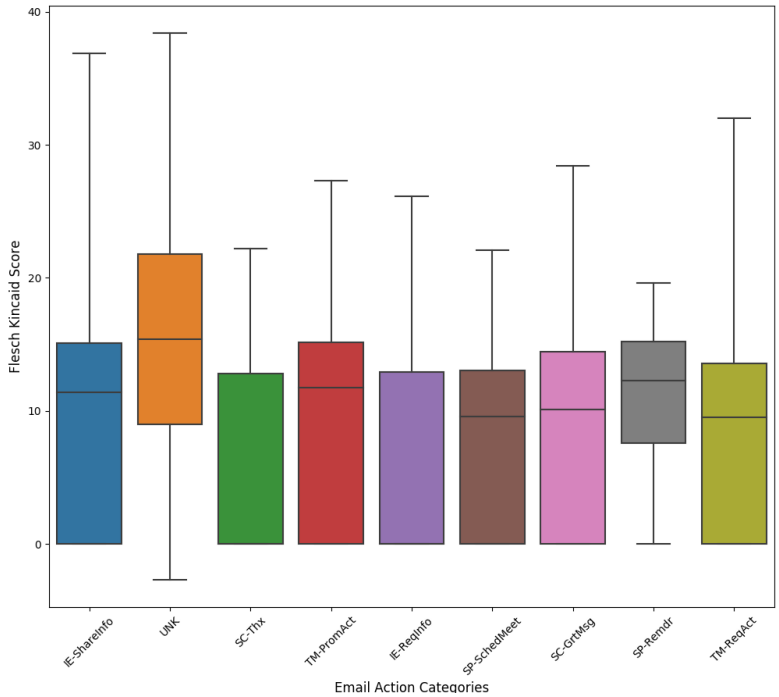
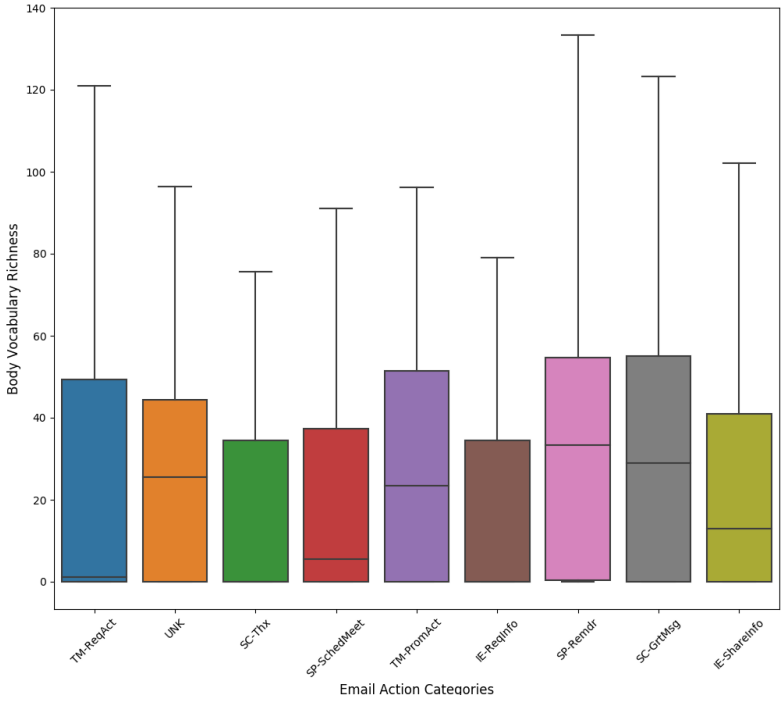



Figure 11: Inter-Annotator Correlation across email act subcategories for annotated Avocado corpus

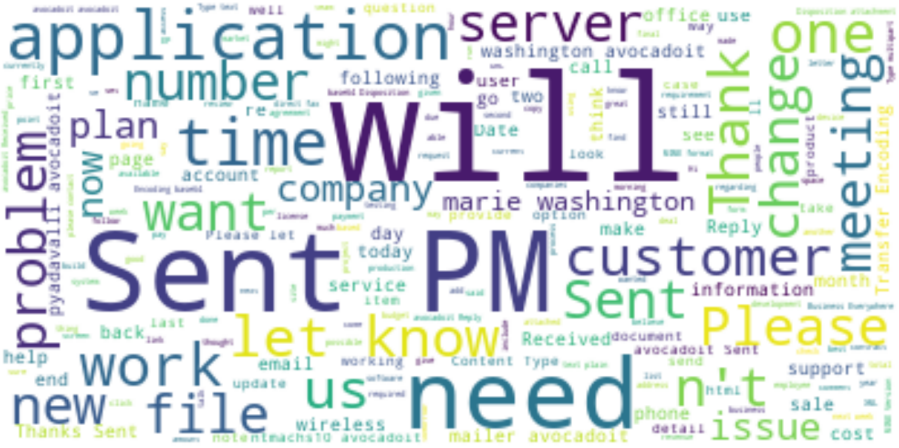



(a)




(b)

Figure 12: Boxplot for Avocado emails for (a) FK score. (b) Vocabulary Richness.




(a)


(b)

Figure 13: Word cloud for Avocado emails under (a) *Share Information (IE-ShareInfo)*. (b) *Requesting Action (TM-ReqAct)* subcategory

# 4 Text Generation with Word-level RNNs

We start with the first architecture for text generation, the word-level recurrent neural network for text generation discussed in Section 3.3.1. We use it to generate two types of text – stories and emails. Here, we outline the experimental conditions used to build and train it and study its performance. The quantitative analysis of the samples generated includes a metric review with correlation and ranking, followed by the results of the top metrics on the datasets. The qualitative analysis includes samples of the text generated by this model.

## 4.1 Experimental Setup

The architecture was developed in Python 3.6 using Keras (Version 2.2.4) and TensorFlow (Version 1.11.0). The network unit used is a Bidirectional LSTM (Bi-LSTMs) [18]. The model is built using 128 hidden LSTM units and the Bi-LSTM has two layers of LSTM units – therefore, 256 hidden units in total were used for training. The input text content needs to be fed into the RNN in the form of words. We use the $cross-entropy$ or $softmax$ optimization technique [36] to compute the training loss, the *Adam* optimization technique [55] is used for updating weights. We use the greedy sampling $argmax$ based technique (Section 3.4) to sample words from the wRNN language model during generation. We explain the hyperparameter selection steps for the architecture along with the sample generation setup for stories and emails in Sections 4.1.1 and 4.1.2 respectively.

### 4.1.1 Generating Stories

**Architecture Setup.** The training hyperparameters were chosen using the GridSearch[32] technique provided by the Scikit-Learn library. We tuned the networks for batch sizes 2, 4, 8 and epochs of 100, 1000 and 10,000. We also experimented with hidden unit sizes of 128 and 256. We manually selected the unrolling size length by experimenting with values between 15 to 25 in increments of 5 units. Similarly, the learning rate ($10^{-2}$) was selected after running the algorithm with learning rates of $10^{-3}$, $10^{-2}$ and $10^{-1}$. We considered an unrolling size or sequence length of 20 for training,

[32] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

i.e., the network looks back up to a sequence of 20 words to predict the next probable word, i.e., a sequence step of 1. Among the other hyperparameters, we considered a batch size of 4 with a learning rate of $10^{-2}$. The word-level RNN (**wRNN**) was trained for a total of 10,000 epochs on the wP-256 and wP-512 datasets for building the language model for story generation. All the experiments were conducted on a server with 4 Tesla M10 GPUs using CUDA (Version 9.1.85) with a 3.20GHz Xeon CPU E5-2667 and 512 GB of memory. The wRNN language model was trained on the **wP-512** dataset for 10,000 epochs with a batch size 4 and 256 Bi-LSTM hidden units for a total training time of 52 hours.

**Sampling Setup.** The model generated stories of length 150 words. We varied the $softmax$ temperature or $\tau$ from 0.25 to 1.0 in intervals of 0.25. We generated 200 samples for each value of $\tau$, by selecting randomly 200 different prompts from the WritingPrompts test set. As a starting seed, we provided the model with the story prompt. In cases where additional padding text was required (e.g., prompt length less than 20 words), we also added the starting 20 words from the human written story corresponding to the selected prompt.

### 4.1.2 Generating Emails

**Architecture Setup.** As explained above, the experimental setup was kept the same and the hyperparameters were tuned slightly due to the change in the dataset. We experimented with a few combinations for the hyperparameters using the GridSearch algorithm from Scikit-Learn – number of RNN nodes, number of layers, epochs and time steps were chosen empirically. The input data was split into mini batches of 10 and trained for 1000 epochs with a learning rate of $2 \times 10^{-3}$. The sequence length was selected as 20. The system was trained on an Amazon Web Services EC2 Deep Learning instance using an Nvidia Tesla K80 GPU. The training took about 32 hours.

**Sampling Setup.** Email generation is not as straightforward as story generation from given prompts. The main intention behind phishing emails is to lure unknowing victims by sending them benign looking emails embedding with malicious phishing cues. Therefore, to automate the process of targeted phishing email generation, we followed the procedure of **injecting** malicious

intent during training and **generating** malicious content in the synthetic emails. We followed a *percentage based influx* of malicious content into the training model along with the legitimate emails. The training models were built by varying the percentage (5%, 10%, 30%, 50% and 70%) of phishing emails selected from the entire phishing dataset (Table 5) along with the entire legitimate emails dataset. We trained separate RNN models on all these configurations. 100 samples generated by each of these models were used in evaluation – the datasets are termed as *m5, m10, m30, m50 and m70* based on the percentage of malicious content. For each sample we generated the text until the first <EOT> was generated. We also included 100 samples (termed as $L$)) from the wRNN model trained only on legitimate emails from the Enron dataset.

We generated the samples by varying the temperature values to 0.25, 0.5, 0.75 and 1.0. For our evaluation experiments, we randomly select 100 system generated samples for each temperature. As a starting seed, we provided the model with the email subject line and the first line as prompt. In cases where additional padding text was required, we also added the starting 10 words from the email written by the humans from the dataset.

## 4.2 Quantitative Analysis

We divided the analysis of the wRNN into generated story and email samples. The performance of wRNN is reported on a set of metrics that evaluate the semantic and syntactic natures of the text as well as the responsiveness of the generative model.

### 4.2.1 Metric Correlation and Ranking

Studying the inter-metric correlation is important to ascertain the relation and importance of a metric with the other metrics. Through metric ranking, we can understand better the performance of a metric on determining the type of written content – human authored or machine generated. In this section, we study the correlation of the metric scores on the text generated by the wRNN and study how the metrics rank with respect to their ability to differentiate between the author of the textual content.

**Metric Correlation.** The correlation heatmap shows if there exists any dependence between metrics as well as their importance. In certain cases, stronger correlations may help ascertain the importance of some metrics with respect to others, and may show how one class of metric is changing with another. We use the Pearson's Correlation Coefficient ($\rho$) to determine the inter-metric correlation.

We observe strong correlations between the Sentence Connectedness metrics (*sent_conn* and *HM_sent_conn*) in Figures 14 and 15. Also, for the correlation values of the metrics' scores on the stories (Figure 14), moderately high correlation exists among the TTR and DCR metrics and the average sentence length. The figure shows perfect correlation among the percentage of nouns and verbs. For emails, we do not observe much correlation between metrics in Figure 15.

**Metric Ranking.** In this experiment, we use the metric values for each story and the label 'human' versus 'automatic' for the generated samples. The scores of the evaluation metrics are given to the well-known regression-based LASSO algorithm [92]. In this way, we can determine which metrics are better at distinguishing between manual and auto-generated samples. Here, we use 5-fold cross-validation with the linear 'LassoCV'[33] model provided by Python's Scikit-Learn package. Equations 3 and 4 show the regression coefficients for the LASSO models tested on the stories and email evaluation scores, respectively. Here $Y$ denotes the outcome variable, i.e, 'label'. The absolute value of the correlation coefficient determines the level of impact of a unit change in the known variable (here, the evaluation metrics) on the estimated variable (here, the nature of written content – human or machine) and the sign (positive or negative) of the coefficient determines the nature of the impact of the variable. The model iterates over 100 possible alpha values to select the best $\alpha$. For the stories, the observed $R^2$ and adjusted $R^2$ values of the regression model are 4.56% and 2.52% respectively. For emails, the $R^2$ and adjusted $R^2$ values are 7.68% and 5.79% respectively.

Figure 16 shows the ranks of the metrics for generated stories – sentence connectedness metrics are the best, followed by TTR and average sentence length. The results of the metric scores on the

[33]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html


Figure 14: Heatmap showing Pairwise Metric Correlation for Stories generated by Word-RNN

emails are shown in Figure 17. In metric ranking, the absolute values of the regression coefficients are more important to measure how much they are correlated (positively or negatively) with a *label*.[34] We observe that the sentence connectedness metric ($HM\_sent\_conn$) has the highest rank, followed by the readability metrics – Dale-Chall Readability Score (DCR) and Flesch Reading Ease (FRE).

[34]Here, the *author* of the content.

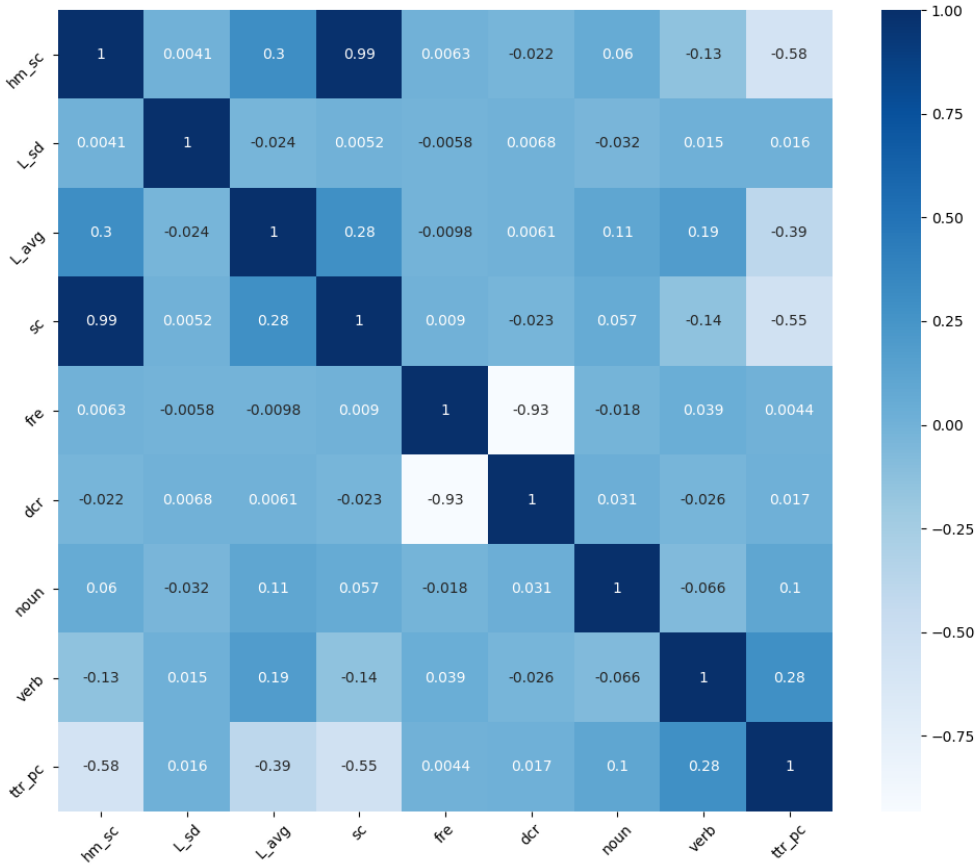



Figure 15: Heatmap showing Pairwise Metric Correlation for Emails generated by Word-RNN

$$\begin{aligned} Y = {} & 0.036 * sent\_conn + (-0.028) * HM\_sent\_conn + 0.015 * ttr\_pc \\ & +(-0.008) * L\_avg + 0.001 * noun + 0.001 * dcr + (-0.0) * fre \\ & +(-0.0) * L\_sd + (-0.0) * biOL + 0.0 * verb \end{aligned} \tag{3}$$

$$\begin{aligned} Y = {} & 0.191 * HM\_sent\_conn + 0.141 * dcr + (-0.131) * sent\_conn \\ & +(-0.028) * L\_sd + 0.024 * ttr\_pc + 0.022 * L\_avg \\ & +0.112 * fre + (-0.04) * verb + 0.019 * noun \end{aligned} \tag{4}$$

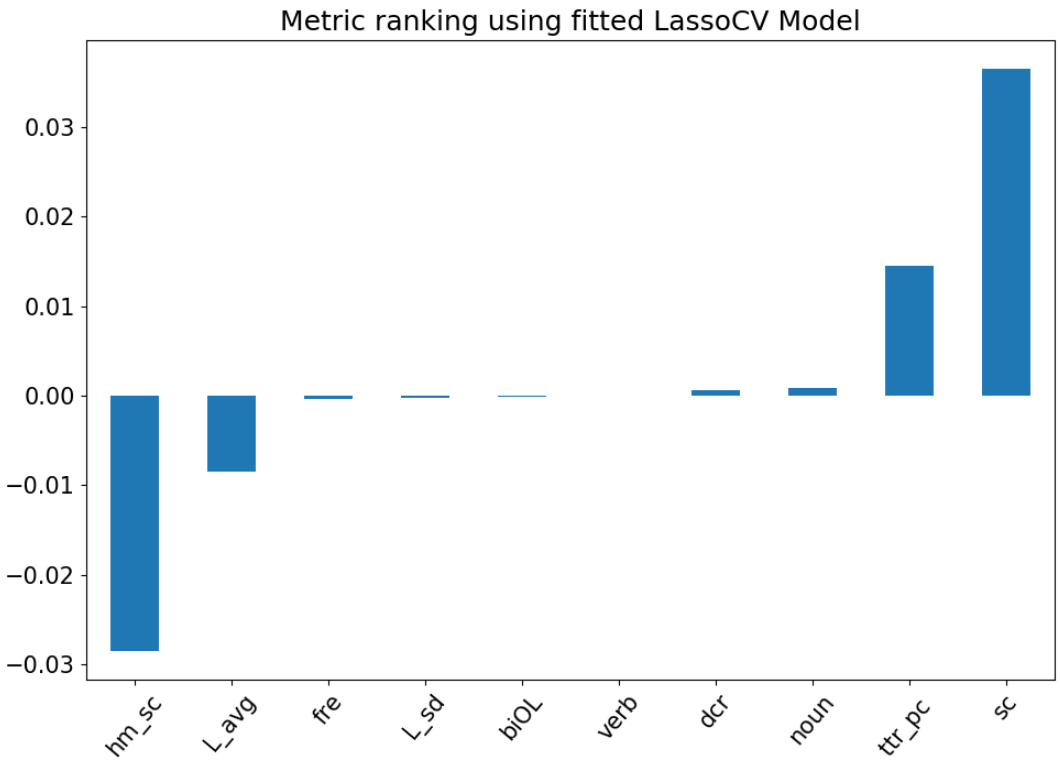



Figure 16: Metric Ranking with Lasso Regularization Method for Stories Generated by Word-RNN

The metric correlation and ranking experiments help us select and study in detail the best set of evaluation metrics and how the trained generative language model performs with respect to human authored content.

### 4.2.2 Analyzing Stories and Emails

In this section, we report the observations on the relatedness and readability, semantic quality and syntactic nature of generated content using the metrics described in Section 3.5. These metrics review the bigram overlap between the prompt and the generated text, the complexity of the syntactic properties like sentence length and uniqueness of words, and sentence-level coherence measured using the sentence connectedness metric.

**Story-Prompt Relatedness and Email Readability.** Conventional models often fail to produce text that is semantically and contextually related to a given prompt or starting seed. A

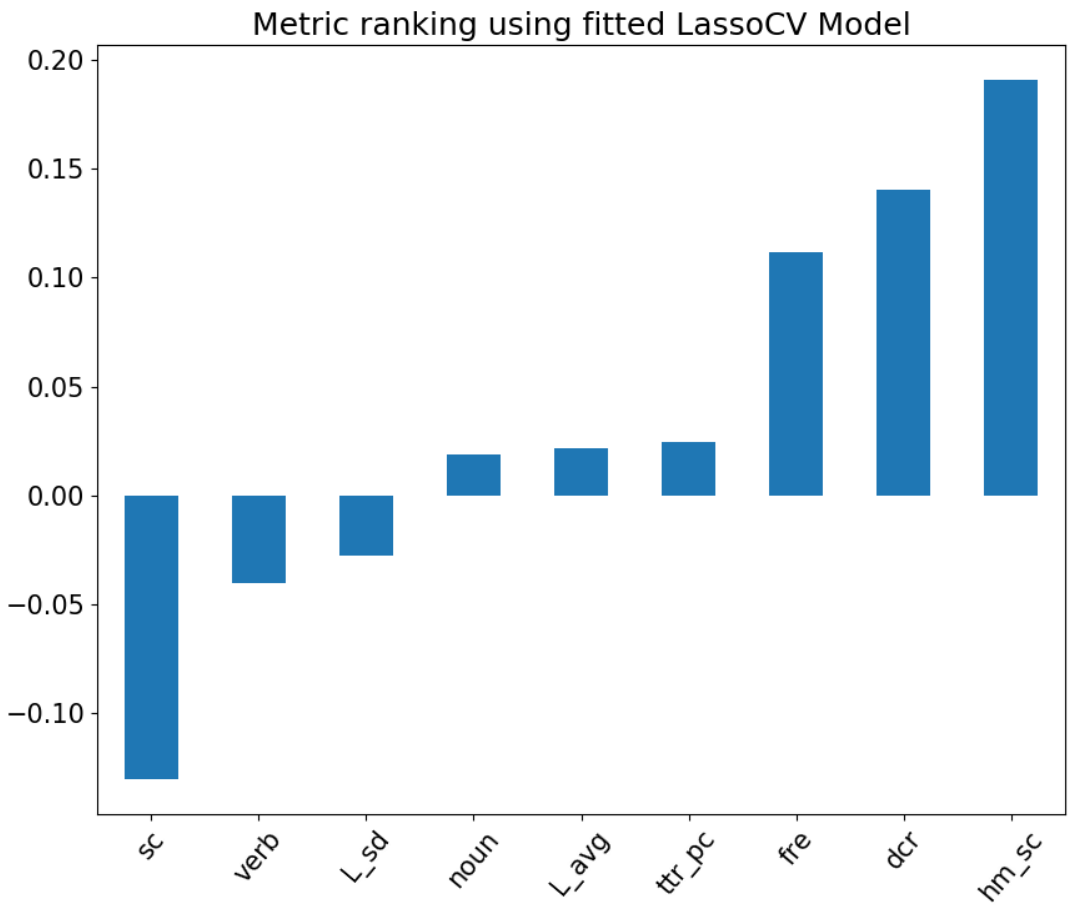



Figure 17: Metric Ranking with Lasso Regularization Method for Emails Generated by Word-RNN

provided starting seed acts as a guide for the generative model by providing prior context for it to choose the best possible sequence of words and/or phrases from the probabilistic distribution of the trained language model. Here, we report wRNN's performance in terms of $n$-gram based overlap with the given story prompt. We report for $n = 2$, i.e., the mean **bigram overlap percentage** scores with the variation in sampling temperature from 0.25 to 1.0, with steps of 0.25. Figure 18 shows the trend in *biOL* overlap percentage scores for the story samples generated by the Word-level RNN with respect to human scores.

The prompt-based conditioning for the stories yielded prominent results but the same for emails was not worthwhile since the email prompts are subjects which in multiple cases were missing or too

short ($\leq 3$ words) to compare with. Therefore, we evaluated the generated email samples based on the scores of the readability metric, **Flesch-Kincaid Grade (FKG) level** [31]. The variation in the FKG scores of the generated email samples is shown in Figure 19. For emails, the samples were generated using different datasets – *m70* to *m10* are samples generated by varying the malicious content influx percentage and $L$ denotes the emails generated by training only on legitimate benign emails from Enron. The figure shows that for emails, the lower the malicious content, the closer to the human readable scores the values are.

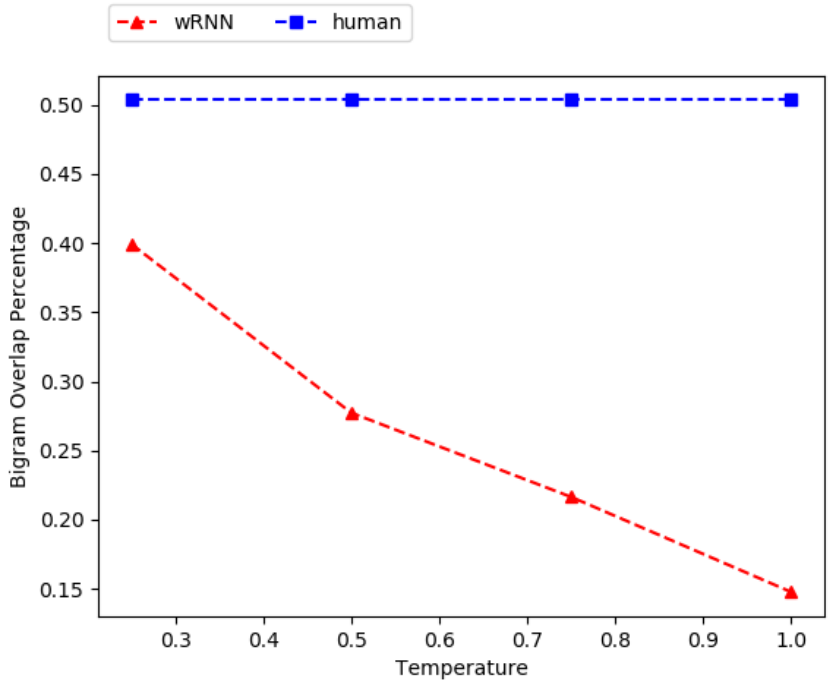



Figure 18: Comparing wRNN for generated stories for *biOL* by varying temperature ($\tau$).

**Syntactic Style and Complexity – S.D. of Sentence Length.** We study the average or mean and the standard deviation (std. dev.) of sentence lengths. Prior metric importance studies [20] have revealed that $L_{sd}$ is a good metric and has a positive correlation with the nature of the textual content being evaluated (human or machine-written) for stories. The $L_{sd}$ values of the story samples generated by the Word-RNN model (wRNN) at varying temperatures are shown in Figure 20 and for generated emails in Figure 21. Although $L_{sd}$ is not a higher ranked metric for analyzing wRNN stories, Figure 17 shows that it has a moderately high negative correlation for

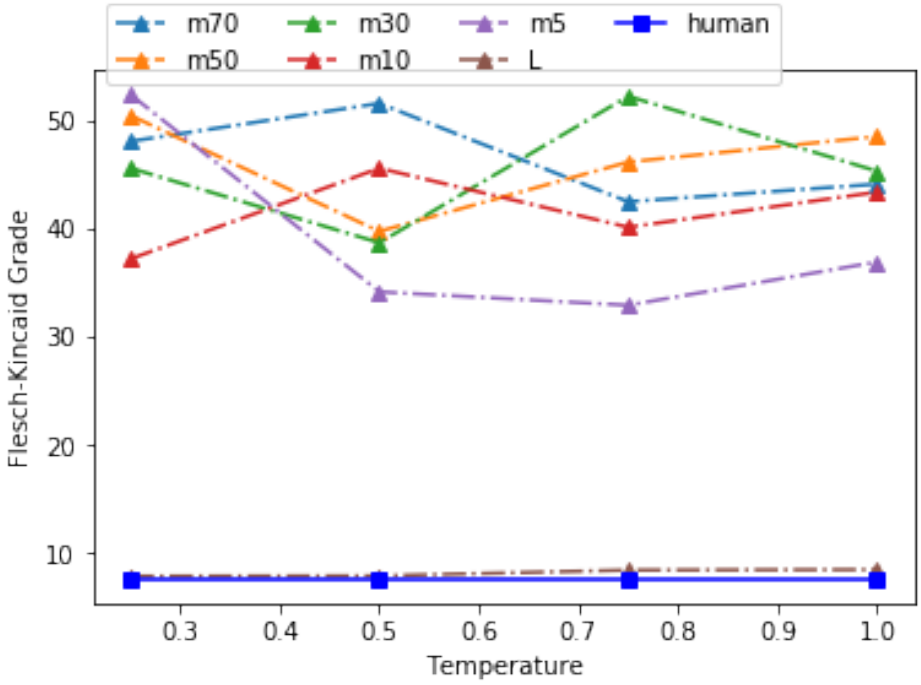



Figure 19: Comparing wRNN for generated emails for FKG by varying temperature ($\tau$).

emails.

In Figure 20 for stories, the trend of the scores are non-uniform with the closest values to human scores being recorded at $\tau = 0.25, 0.75$. The non-deterministic higher sampling value makes the trained model generate incoherent and longer sequences of text causing more variation in the sentence length. At lower $\tau$, the lower $L_{sd}$ scores show that the model might just be repeating the same phrase or sequence of words leading to low variation. For emails, the samples generated by lower percentages of malicious (m5, m10) have a consistent variation in $L_{sd}$ trend shown in Figure 21. The model trained only on legitimate samples has considerably close scores to human written emails.[35]

**Syntactic Style and Complexity – Type Token Ratio.** The Type Token Ratio (TTR) measures sentence complexity, lexical richness or variety in vocabulary of the generated content. Studying the TTR values of the generated gives an insight into the nature of the generated content based on the class of the generative model. The performance of the wRNN model on story datasets

[35]T-test of means shows a $p-$value of 0.243 where $\alpha = 0.05$

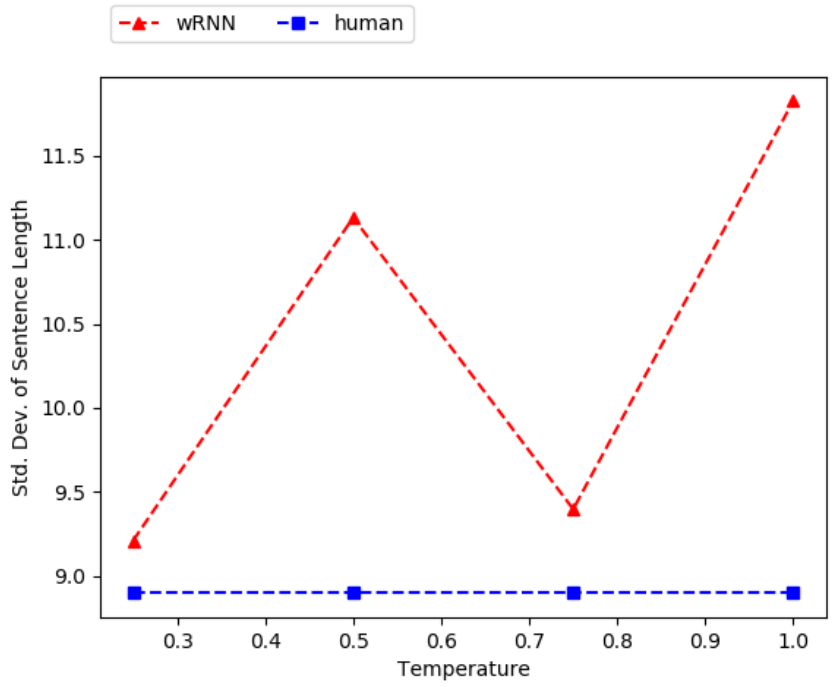



Figure 20: Comparing wRNN for generated stories for $L_{sd}$ by varying temperature ($\tau$).

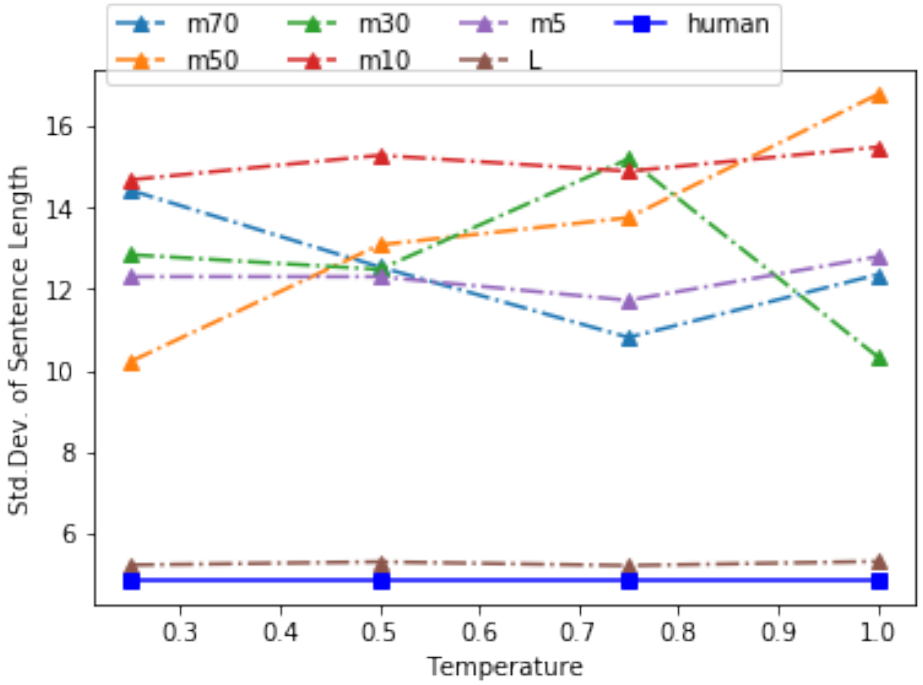



Figure 21: Comparing wRNN for generated emails for $L_{sd}$ by varying temperature ($\tau$).

in terms of TTR values is shown in Figure 22. It shows an increasing value with an increase in softmax temperature. The changes in the TTR values for the different email models are shown in Figure 23. As seen earlier, TTR scores for the generated stories are closer to the humans as the $\tau$ increases. For emails, the lower values of malicious content in the emails, the closer the scores are to the scores on the human writing. Lower values mean less uniqueness, so more injection of malicious data reveals that the generated samples will be further away in nature from human written emails.

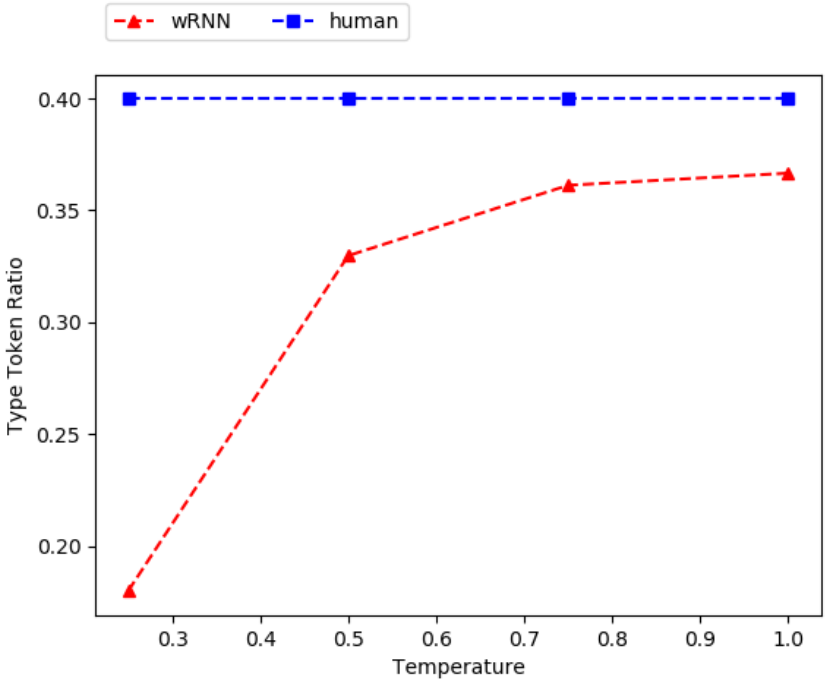



Figure 22: Comparing wRNN for generated stories for TTR by varying temperature ($\tau$).

**Harmonic Mean of Sentence Connectedness.** This metric is for cohesion at the sentence level. Using Sent2Vec-based embeddings and a pre-trained Wikipedia-based Bigram model,[36] each sentence token is vectorized. The angular variation between sentence vectors is calculated in radians – we calculate the mean ($\mu$) and standard deviation ($\sigma$) of the total set of variations. Finally, the harmonic mean (H.M.) of the mean and standard deviation of the sentence connectedness values is calculated ($HM\_sent\_conn$). Lower values of harmonic mean indicate less angular variation and deviation between vectors.

[36]https://github.com/epfml/sent2vec

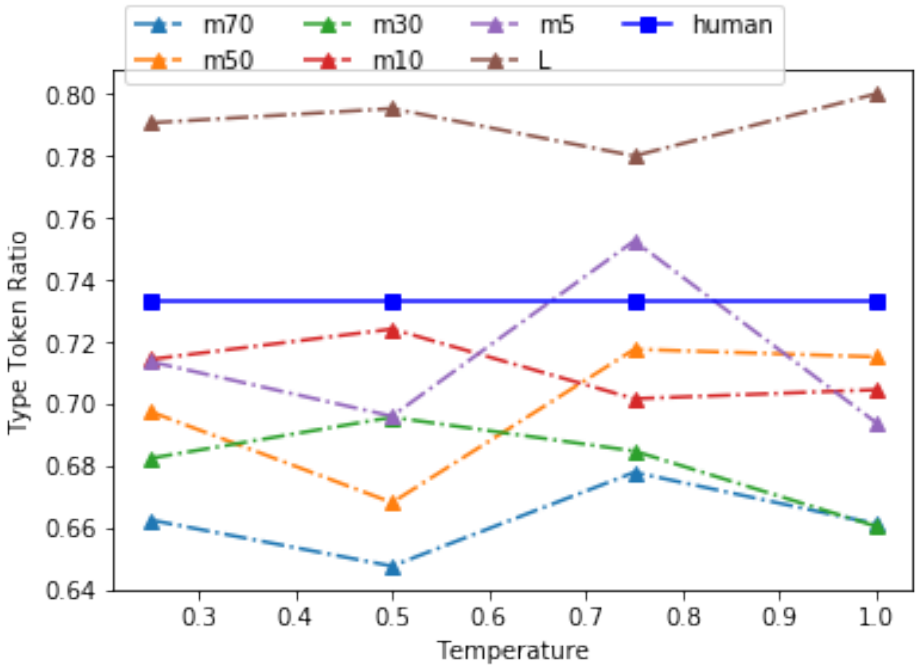



Figure 23: Comparing wRNN for generated emails for TTR by varying temperature ($\tau$).

The performance of the wRNN model on story datasets in terms of sentence connectedness scores is shown in Figure 24. For the generated stories, with an increase in the $\tau$ value, the metric's values become closer to the human scores, with the closest values at $\tau = 0.75$. The metric's scores on the emails generated by the wRNN models trained are shown in Figure 25. It shows that the model trained only on legitimate instances score significantly closer to the human scores of $HM\_sent\_conn$.[37]

### 4.2.3 Detection Algorithm for Emails

**Phishing Detection Algorithm.** We trained text classification models using Support Vector Machines (SVM), Naive Bayes (NB), and Logistic Regression (LR) models on a training data of 300 legitimate emails from WikiLeaks archives[38] and 150 phishing emails from Cornell PhishBowl [50]. We tested the data on 100 legitimate emails from WikiLeaks archives that were not included in the training set and 25 'fake' emails that were generated by the natural language generation model

[37] T-test scores of mean reveals $p-$value=0.456 with $\alpha$=0.05

[38] `https://wikileaks.org/`

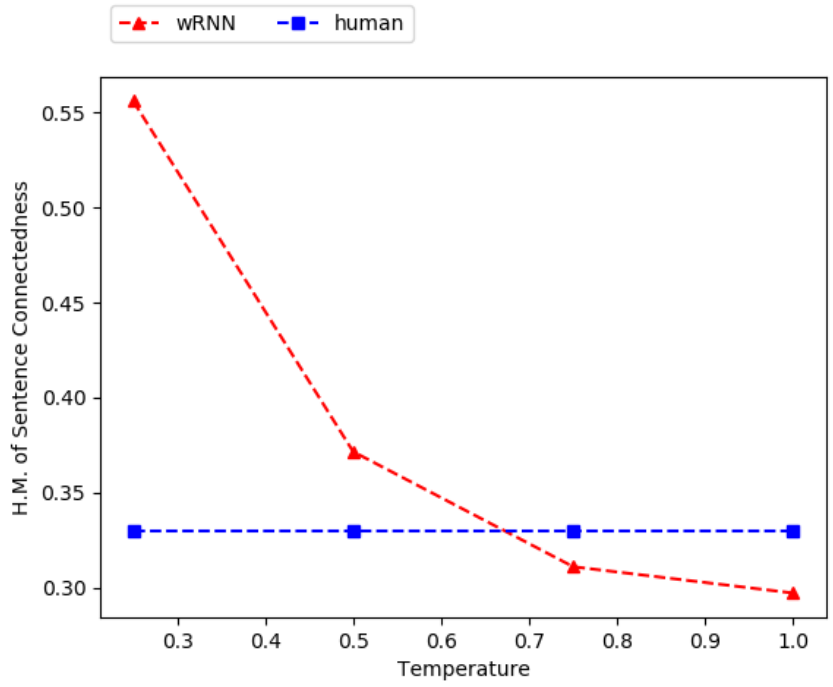



Figure 24: Comparing wRNN for generated stories for $HM\_sent\_conn$ by varying temperature ($\tau$).

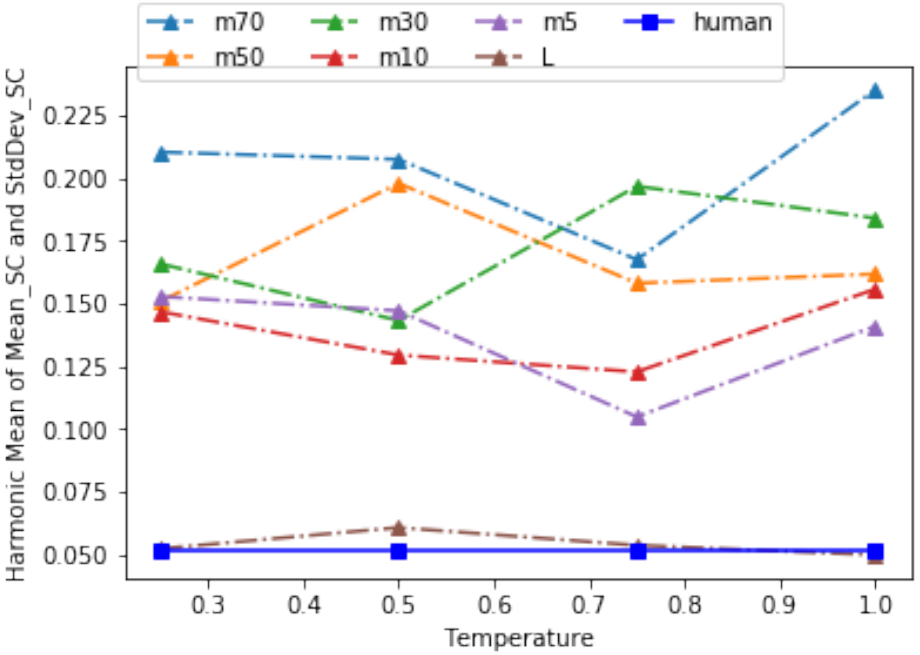



Figure 25: Comparing wRNN for generated emails for $HM\_sent\_conn$ by varying temperature ($\tau$).

trained on a mix of legitimate and 50% malicious emails. We randomly selected the emails (the distribution is: 2 samples generated at a temperature of 0.2, 10 samples at temperature 0.5, 5 samples at a temperature of 0.7 and 8 samples at temperature 1.0) for evaluation.

We used the Scikit-Learn Python library to generate the *document-term matrix* and the *word count vector* from a given sample of email text body used as a feature for training the classification models. Table 7 reports the accuracy, precision, recall, and F1-scores on the test dataset using SVM, Naive Bayes, and Logistic Regression classifiers. Despite the incoherent nature of the generated emails, the text-based classifiers do not achieve a 100% accuracy as well as F1-scores.

Table 7: Classification metrics (in percentage) of generated emails

| **Classifier** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| SVM | 71 | 72 | 85 | 78 |
| NB | 78 | 91 | 75 | 82 |
| LR | 91 | 93 | 95 | 94 |

**Baseline Comparison.** The authors in [7] discuss using a Recursive Transition Network for generating fake emails similar in nature to legitimate emails. The paper discusses a user study testing the efficacy of these fake emails and their effectiveness in being used for deceiving people. The authors use only legitimate emails to train their model and generate emails similar to their training data - termed as 'fake' emails. In this section, we compare a couple of examples selected randomly from the emails generated by the Dada Engine used in [7] and the output emails from the deep neural learner in Box 4.1. The examples provide evidence that emails generated by the RNN are more on the lines of phishing emails than the emails generated by the Dada Engine. Of course, the goal of the email generated by the Dada engine is masquerade, not phishing. Because of the rule-based method employed that uses complete sentences, the emails generated by the Dada engine have fewer problems of coherence and grammaticality.

## 4.3 Qualitative Analysis

As part of the qualitative analysis we include samples generated by the trained story and email-based language models.

**Box 4.1: Samples comparing wRNN and Dada engine.**

**Samples generated by the Word-RNN**
**Example I:** Hi will have temporarily information your account will be restricted during that the Internet accounts and upgrading password An data Thank you for your our security of your Account Please click on it using the $< NET >$ server This is an new offer miles with us as a qualified and move in
**Example II:** Sir Kindly limit, it [IMAGE] Please contact us contained on this suspension will not be = interrupted by product, or this temporary cost some of the
**Sample generated by the Dada Engine**
Great job on the op-ed! Are you going to submit? Also, Who will be attending?

### 4.3.1 Generated Samples of Stories

For each of the selected test prompts, we randomly selected a human-written story from the test set and use the entire 1000 word story for the comparison during the evaluation step. We include sample examples of generated stories from prompts in this Section. The qualitative analysis shows samples generated at selected temperatures. At lower temperatures, we see the repetitiveness in the generated text. The coherence in the generated content is more coherent with an increase in the temperature.

Apart from listing the samples generated by the language models at different temperatures for different prompts, we also present story samples generated by the wRNN model for one given prompt at different temperatures in Table 8 for a better comparison. We also mention the generated samples at different temperatures in the Tables 9, 10 and 11.

### 4.3.2 Generated Samples of Emails

We performed a qualitative analysis of the samples generated using the different trained RNN language models. Below are the examples of the email bodies generated with various training models, depending on the amount of legitimate and malicious (phishing) emails used for training, by varying the temperature or $\tau$ value in the sampling phase. This is followed by accuracy results of a detection algorithms (trained classifiers) on a subset of these generated email bodies (after

Table 8: An example showing a prompt and the stories generated using the **trained wRNN** model at different $\tau$

| **Prompt** | Whilst restocking a vending machine, a janitor finds something horrific jammed inside. |
|---|---|
| **word-based RNN** $t = 0.25$ | with his worst nightmare . the machine had n't been open for 13 years the elders said. , you you you you you , you you you together you you . you you , , you you you , , one it you you , [...] |
| **word-based RNN** $t = 0.5$ | with his worst nightmare . the machine had n't been open for 13 years the elders said. , mashed of if you you a together personally - , , patches you together, together you , together , personally - together 's if , it would you like twice of together were if [...] |
| **word-based RNN** $t = 0.75$ | with his worst nightmare. the machine had n't been open for 13 years the elders said . of flesh ? , . , you you . jackets took - like flesh you before you gritty together on in before you you bunched , together hand a patches together between of [...] |
| **word-based RNN** $t = 1.0$ | with his worst nightmare. the machine had n't been open for 13 years the elders said. infested bug you one were dead of why not think parking asphalt - cotton 's infested you to between . on , in you and you on personally together, you you n't you think coat asphalt you [...] |

Table 9: An example showing a prompt and a generated story using the **trained wRNN** model at $\tau = 0.25$

| **Prompt** | Write a horror short where the people inside are not complete idiots. |
|---|---|
| **word-based RNN** $t = 0.25$ | " mommy ! ! *mommy ! , it you you you you , , you you you you you you it you , , you . - you , you you you you you you you bug , you , you you you it you you you you [...] |

Table 10: An example showing a prompt and a generated story using the **trained wRNN** model at $\tau = 0.75$

| **Prompt** | Share an excerpt from a story you've been writing in your head |
|---|---|
| **word-based RNN** $t = 0.75$ | thinking you run your sword through him as you lift your shield arm and feel an impact against it. you not tore infested not someone it hand personally it, personally flesh,, someone and you one you between hand together- , dead a you on you together of, together field together twice it you tore n't - flat flat from you old to asphalt and it not field from you should like - together twice together the one infested lot infested old ? [...] |

slight post processing, e.g., removal of unicode or special characters,[39] sentence fragments, nonsensical strings or substrings). Thus, this section gives an insight into the qualitative review of the generated emails.

[39]https://www.rapidtables.com/code/text/unicode-characters.html

Table 11: An example showing a prompt and a generated story using the **trained wRNN** model at $\tau = 1.0$

| **Prompt** | Love is a chemical reaction, and you have the formula. |
|---|---|
| **word-based RNN** $t = 1.0$ | Years of being mocked for my beliefs, now i have proof that love<br>is simply a chemical reaction.., a n't would dead infested if infested coat together together,<br>were it coat it dead parking bug think to it from flesh dead -.<br>why patches parking it , think of would the not - - bunched it 's together took . - together you you<br>- between of bunched personally from would it think personally between asphalt tore ?<br>[...] |

**(A) Trained using legitimate emails only.** The examples show that while some parts of the generated content are readable, the sequence of text fragments generated make little sense when read as a whole. When comparing these with the phishing email structure described in [24], the generated emails have very little malicious content. There exists some incongruous text pieces that highlight the incoherent nature of the content. The examples of emails generated using models trained on legitimate emails are shown in Box 4.2. These samples however, are close to human written emails.

**Box 4.2: Samples by training on legitimate data only**

**Example I at $\tau$ = 0.75:** Sir I will really see if they were more comments tomorrow and review and act upon this evening <NET>. The engineer I can add there some <LINK> there are the issues <NET>. Could you give me a basis for the call
**Example II at $\tau$ = 1.0:** Dear <NME> The article in the <NME> offers promotion should be somewhat changed for the next two weeks. <NME> See your presentation today.

**(B) Trained using legitimate + 5% of malicious emails.** For the first step, the model is built by training on all the legitimate emails and 5% of the phishing email instances. Thus, for this model, we create the input data with 603 legitimate emails and 114 randomly selected phishing emails. The model thus consists of benign and malicious emails in an approximate ratio of 5:1. Some intent and urgency can be seen in the email context. But the incoherent words remain. We show as examples two samples generated using varied temperatures in Box 4.3.

**(C) Trained using legitimate + 30% of malicious emails.** We further improve upon the model proposed in (B). In this training step, we feed the text generator all the legitimate emails (603 benign) coupled with 30% of the malicious emails data (683 malicious). This is an almost

**Box 4.3: Samples by training on legitimate + 5% of malicious emails.**

**Example I at Temperature = 0.5:** Sir Here are the above info on a waste of anyone, but an additional figure and it goes to <NET>. Do I <NET> got the opportunity for a possible position between our Saturday <NME> or <NET> going to look over you in a presentation you will even need <NET> to drop off the phone.
**Example II at Temperature = 0.75:** Hi owners <NET> your Private <NET> email from <NET> at <NET> email <NET> **Information I'll know our pending your fake check to eol** thanks <NET> and would be In maintenance in a long online demand.

balanced dataset of benign and phishing emails. The examples in Box 4.4 demonstrate the variation in text content in the generated emails. We see the presence of some malicious textual cues [24] in the generated emails.

**Box 4.4: Samples by training on legitimate + 30% of malicious emails.**

**Example I at Temperature = 0.5:** Sir we account access will do so may not the emails about the <NET> This <NET> is included at 3 days while when to <NET> because **link below to update your account until the deadline** we will received this information that we will know that your <NET> account information needs.
**Example II at Temperature = 1.0:** Dear registered **secur= online**, number: hearing from This trade guarded please account go to pay it. To **modify your Account then fill in necessary from your notification preferences,** please PayPal account provided with the integrity of information on the Alerts tab.

**(D) Trained using legitimate + 50% of malicious emails.** In this training step, we consider a total of 50% of the malicious data (1140 phishing emails) and 603 legitimate emails. This is done to observe whether training on an unbalanced data, with twice the ratio of malign instances than legitimate ones, can successfully incorporate obvious malicious flags like poisonous links, attachments, etc. We show two examples of emails generated using deep learners at varying sampling temperatures in the box 4.5. The generated text reflects malicious features like URL links and tone of urgency. We can assume that the model picks up important cues of malign behavior. The model then learns to incorporate such cues into the sampled data during training phase.

**Box 4.5: Samples by training on legitimate + 50% of malicious emails.**

**Example I at Temperature = 0.75:** If you are still online. genuine information in the message, notice your account has been frozen to your account in order to restore your account as click on CONTINUE Payment Contact <LINK> UK.
**Example II at Temperature = 0.5:** Hi will have temporarily information your account will be restricted during that the Internet accounts and upgrading password An data Thank you for your our security of your Account **Please click on it using the <NET> server** This is an new offer miles with us as a qualified and move in

### 4.3.3 Error Analysis

We review *two types of errors* observed in the evaluation of the RNN text generation models developed in this study. *First*, the text generated by multiple RNN models suffers from **repetitive tags and words**. The example of the email body below in Box 4.6 demonstrates an incoherent piece of text generated by the RNN trained on legitimate emails and 50% of phishing emails with a temperature of 0.5, characterized by repetitive generation of the same word unit. Possible reasons include training setup of the Bi-LSTM network, or because of the relatively small training dataset we have used. A third issue could be the temperature setting. More experiments are needed to determine the actual causes.

**Box 4.6: Email samples for Error Analysis**

**Generated sample at $\tau = 0.5$ with repetitive words**
Hi 48 PDX Cantrell <LINK> **<NET> <NET> ECT ECT <NET> <NET> ECT ECT <NET> <NET> ECT ECT <NET> <NET> ECT ECT <NET> F <NET> ECT ECT <NET> G Slaughter 06 07 03 57 DEVELOPMENT 06 09 2000 07 01 <NET> <NET> ECT** ENRON 09 06 03 10 23 PM To **<NET> <NET> ECT ECT cc <NET> <NET> ECT ECT** Subject Wow Do not underestimate the employment group contains Socal study about recession impact <NET> will note else to you for a revised Good credit period I just want to bring the afternoon <NET> I spoke to <NET> Let me know if

The *second aspect* of error analysis is to review the misclassification by the statistical detection algorithms. Here we review a small sample of emails that were marked as legitimate despite being fake in nature. We try to investigate the factors in the example sample that can explain the misclassification errors by the algorithms.

Examples (A), (B) and (C) shown in box 4.7 are emails generated from a model trained on legitimate and 50% of phishing data using a temperature of 0.7. There can be quite a few reasons for the misclassification - almost all the above emails despite being 'fake' in nature have considerable overlap with words common to the legitimate text. Moreover, Example (A) has lesser malicious intent. And the amount of malicious intent in Example (B), although notable to the human eye, is enough to fool a simple text-based email classification algorithm. Example (C) has multiple link tags implying possible malicious intent or presence of poisonous links. However, the position of these links plays an important role in deceiving the classifier. Most phishing emails have links at the end of the text body or after some action words like *click*, *look*, *here*, *confirm* etc. In this case, the links have been placed at arbitrary locations inside the text sequence, therefore making it harder to detect. These errors can be eliminated by human intervention or by designing a more sensitive and sophisticated detection algorithm.

**Box 4.7: Email samples for Error Analysis**

**Example (A):** Hi GHT location <EID> Inc Dear <NET> Password Location <NET> of <NET> program We have been riding to meet In a of your personal program or other browser buyer buyer The email does not commit to a secure F or security before You may read a inconvenience during Thank you <NET>

**Example (B):** Sir we account access will do so may not the emails about the <NET> This <NET> is included at 3 days while when to <NET> because the link below to update your account until the deadline we will received this information that we will know that your <NET> account information needs

**Example (C):** Sir This is an verificati= <LINK> messaging center, have to inform you that we are conducting more software, Regarding Your Password : <LINK> & June 20, 2009 Webmail Please Click Here to Confirm

# 5 Text Generation with Transformer-based Language Models

In this chapter, we describe in detail the experimental setup for generating content (stories and emails) using the massive transformer architecture-based language models like OpenAI's GPT [72, 73] and Google's Transformer-XL and XLNet [107, 19]. We discuss the nature of the generated text through the quantitative and qualitative evaluation of the generated samples. As part of the quantitative analysis, we also provide insight into the metric ranking and inter-metric correlation.

## 5.1 Experimental Setup

We discuss the experimental conditions to reproduce the pre-trained transformer based language models as well as fine-tuning the model for text generation. The setup also includes details about sample generation.

**Reproducing Pre-trained Language Models.** We apply the language models explained in Section 3.3.2 for the story generation task in a 'zero-shot' setting. For reproducing the 'massively' pre-trained language models – **OpenAI's GPT and GPT2** and **Google/CMU's XLNet and Transformer-XL**, we used the HuggingFace repository [105] which has implementations of these models for language generation.[40] We ran the models using the PyTorch 1.2.0 framework and Python 3.7.4 on a system with NVIDIA Tesla M10 GPU. We evaluated a total of seven pre-trained language models for this task.

**Fine-tuning Language Models.** Retraining the pretrained models is necessary to condition the model on the domain dataset and make the retrained model generate stylistically and linguistically better textual content. The model fine-tuning can be termed similar to building a language model on the training dataset (emails or stories), where each sequence separated by the delimiter token – $<|$`endoftext`$|>$. We used the Python implementation of the GPT2 models made available by OpenAI.[41] For each dataset, we included the details of the model hyperparameters and training in Sections 5.1.1 and 5.1.2, respectively.

[40] `https://github.com/huggingface/transformers/tree/master/examples\#language-generation`
[41] `https://github.com/nshepperd/gpt-2`

**Sample generation.** The text instances were generated using the top-$k$ sampling technique [27, 73] during the generation process. For each of the pre-trained language models, the instances were generated by changing the hyperparameters, by varying the softmax temperature parameter $\tau$ (where $\tau = \{0.25, 0.5, 0.75, 1.0\}$), while keeping the top-$k$ sampling parameter $k$ constant, where $k \in \{0, 10, 50, 250, 1000\}$.

### 5.1.1 Generating Stories

We explain the setup for reproducing the massive language models in a zero-shot setting and with fine-tuning in this section to generate stories from writing prompts.

We **reproduced the pre-trained language models** following the steps highlighted above for story generation from the prompts. The **fine-tuning experiment** on the different BPE-tokenized versions of WritingPrompts dataset resulted in four models, the hyperparameters and the model training times and average loss achieved on the validation dataset are given in Table 12. Each prompt and story pair is regarded as one instance separated by the delimiter token – $<|$`endoftext`$|>$. The batch size and initial learning rate for the fine tuning experiments were chosen as 2 and $2 * 10^{-5}$. The batch size and the learning rate[42] were chosen based on the computation capability of our GPU. The models were trained using Python 3.6 on a Quadro P1000 GPU. For the four models, *per-word perplexity* using average loss ($\overline{loss}$), i.e., if we consider it equal to $e^{\overline{loss}}$, falls in the range of 13.06 to 18.7 units which is less than for the baseline **fusion** model mentioned earlier [27] on the validation datasets.

Table 12: Hyperparameters for fine-tuning the GPT2-medium and GPT2-large models

| | model params | layers | training epochs | average loss | training time (hrs) |
|---|---|---|---|---|---|
| **wP_512_117M** | 117M | 12 | 10,000 | 2.77 | 40.6 |
| **wP_1024_117M** | 117M | 12 | 10,000 | 2.68 | 75.7 |
| **wP_512_355M** | 355M | 24 | 10,000 | 2.48 | 139.5 |
| **wP_1024_355M** | 355M | 24 | 10,000 | 2.57 | 107.8 |

**Sample generation.** The stories in the test set of the WritingPrompts dataset are used

[42]Initial learning rate with an exponential decay and a decay rate of 0.96 and 10,000 steps

as the references for comparing human written text (**human**) and auto-generated samples. We generated samples by randomly selecting 200 different prompts from the test set. To compare performance with previous open-ended story generation literature, each generated story is limited to 150 words as mentioned in [27, 79]. For each of the selected test prompts, we randomly selected a human-written story from the test set and use the first 150 words from the story for comparison during the evaluation step.

### 5.1.2 Generating Emails

We reproduced the pre-trained language models for email generation following a similar setup used in story generation in zero-shot setting. The **model fine-tuning** can be termed similar to building a language model on the ENRON and AVOCADO datasets, where each email subject and body is regarded as a one sequence separated by the delimiter token – $<$ |`endoftext`| $>$. We used the Python implementation of the GPT2 models made available by OpenAI.[43]

The fine-tuning experiment resulted in four models – the hyperparameters and the model training times and average loss achieved on the validation dataset are given in Table 13. The hyperparameters for fine-tuning the email datasets were kept similar to the ones used in the story generation task and were chosen based on the computation capability of our GPU. The models were trained using Python 3.6 on a Quadro P1000 GPU. For the four models, *per-word perplexity* using average loss ($\overline{loss}$), i.e., if we consider it equal to $e^{\overline{loss}}$, falls in the range of 4.88 to 5.92 units.

Table 13: Hyperparameters for fine-tuning the GPT2-medium and GPT2-large models

| | model params | layers | training epochs | average loss | training time (hrs) |
|---|---|---|---|---|---|
| **E_128_117M** | 117M | 12 | 10,000 | 1.48 | 55.3 |
| **A_128_117M** | 117M | 12 | 10,000 | 1.55 | 80.7 |
| **E_512_355M** | 355M | 24 | 10,000 | 1.35 | 115.4 |
| **A_512_355M** | 355M | 24 | 10,000 | 1.78 | 120.9 |

The email samples were generated in a similar fashion as done for stories. We generated 100 samples for each pre-trained and fine-tuned language model and compare with 100 email samples

[43] `https://github.com/nshepperd/gpt-2`

from the human authored data – Avocado and Enron.

## 5.2 Quantitative Analysis

The relationship among the metrics based on their scores on the generated instances provides insight on whether they are correlated as well as their relative importance. Metric ranking provides a clearer picture on which evaluation metrics are better at differentiating the nature of the content – human written or machine generated. Using the results of the metric study, we selected the best performers and took a deeper look at the performance of the systems on those metrics for the story and email datasets.

### 5.2.1 Metric Correlation and Ranking

The correlation among metrics based on their performance on the generated text determines the importance of the metric and its relation with the other metrics. The metrics' ability to differentiate between human and language models has been shown through a regression-based analysis.

**Metric Correlation.** We used the mean metric scores on the generated instances across different hyperparameter combinations to compute the correlation between each pairwise metric. We used Pearson's Correlation Coefficient ($\rho$) for this purpose.

Figures 26 and 27 are heatmaps showing the measure of correlation amongst the metrics on the stories and the emails respectively. Note the low level of correlation between the readability metrics (fre and dc) in Figure 26. As expected, there is a high positive correlation ($\rho > 0.9$) among the story-prompt $n-$gram overlap metrics (uniOL, biOL and triOL). Interestingly, these overlap metrics have relatively high positive correlation ($\rho > 0.75$) with the style metrics: average sentence length and the distribution of nouns in the textual content. Other interesting high positive correlations exist between the Dale-Chall Readability (DCR) scores and Type-Token Ratio, and between average sentence length and noun usage. A high correlation between the Dale-Chall Readability score and Type-token ratio shows that the generated content has an increased occurrence of difficult and unique words. In Figure 27, there exists no such correlation among the metrics calculated

on the generated emails. However, there exists a strong positive correlation between the sentence connectedness metrics sent_conn and HM_sent_conn with $\rho = 0.98$.

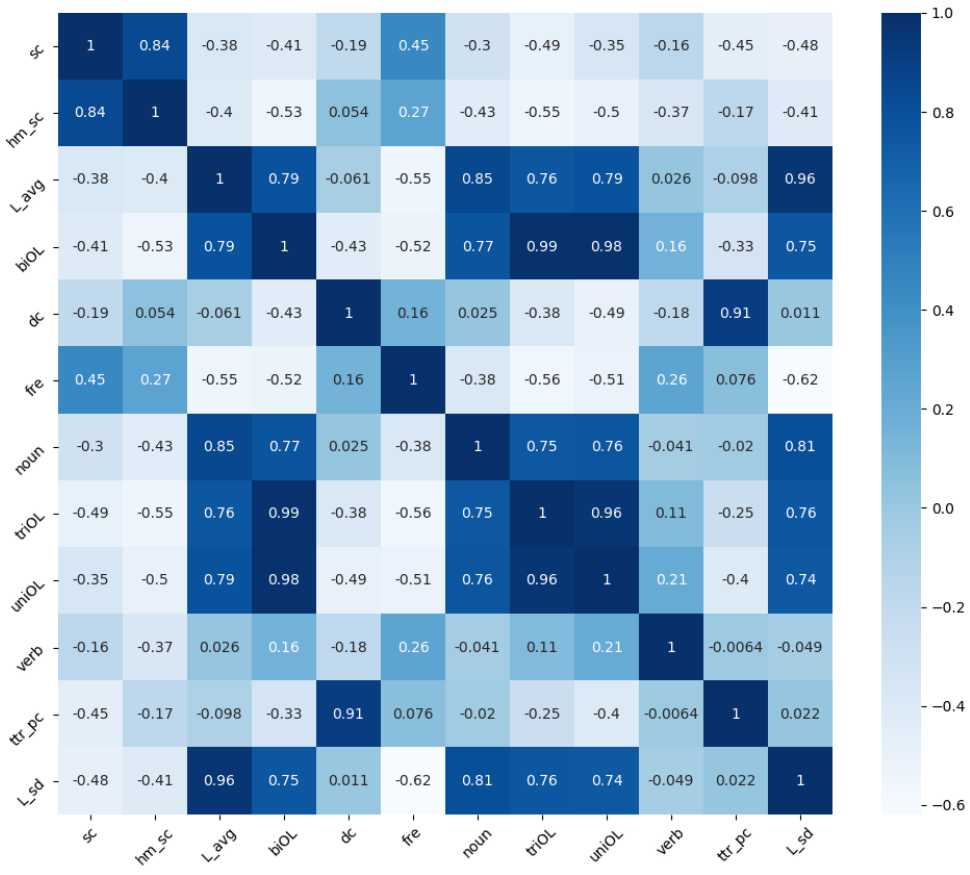



Figure 26: Heatmap showing Pairwise Metric Correlation for Stories

**Metric Ranking.** In this experiment, we use the metric values for each story and the label 'human' versus 'automatic' for the generated samples. The scores of the evaluation metrics are given to the well-known linear regression-based LASSO algorithm [92] along with 5-fold cross-validation analysis. In this way, we can determine which metrics are better at distinguishing between manual and auto-generated samples. As before, we use Scikit-learn's implementation of

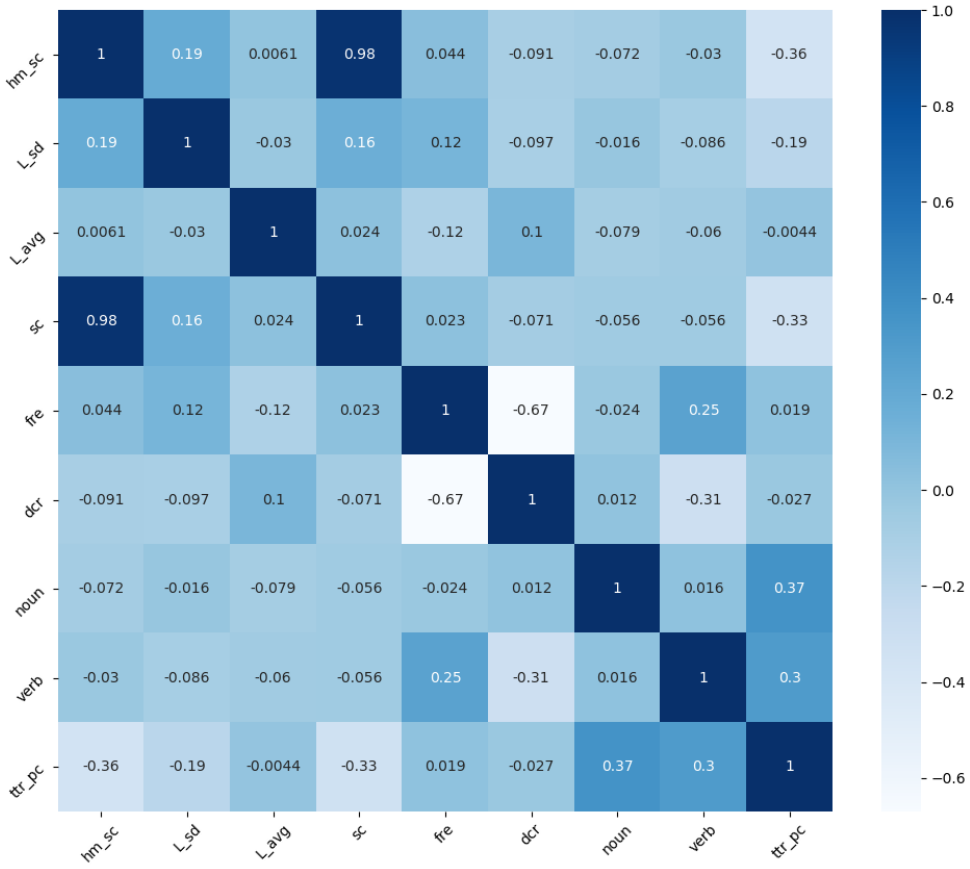



Figure 27: Heatmap showing Pairwise Metric Correlation for Emails

the LASSO algorithm.[44] For the stories, the best $\alpha$ is 0.000651 and the best model score (coefficient of determination, $R^2$) is 84.04%, the adjusted $R^2$ value is 83.78%. For analysis on the emails, the best $\alpha$ was 1.0321 and the best $R^2$ is 4.68% and adjusted $R^2$ is 2.40%.

We see that the metrics capable of giving the most important information for distinguishing the generated samples from human references are $N-$gram overlap percentages between the story and the prompt, specifically bigrams and unigrams. The bigram overlap shows a strong positive coefficient of correlation while unigram overlap percentage has a similar but negative correlation

[44] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html

with the outcome. Here $Y$ denotes the outcome variable, i.e., 'label'. Dale-Chall readability score and Type-token ratio percentage also appear as metrics with strong coefficients for distinguishing between the human and non-human content. The statistical properties of number of words in the textual content at the sentence level like mean sentence length (L_avg) and standard deviation of sentence length (L_sd) are also highly correlated with the label. While mean sentence length has a negative correlation coefficient, standard deviation has a similar but positive correlation with the nature of the content. The measure of interaction of the metrics with the outcome variable in terms of their coefficients based on the fitted linear Lasso model is given by Equation 5:

$$\begin{aligned} Y = 3.511 * biOL + (-2.873) * uniOL + (-1.498) * triOL \\ +(-1.084) * dc + (-0.942) * L\_avg + (0.764) * L\_sd \\ +0.512 * ttr\_pc + (-0.429) * sent\_conn + 0.306 * HM\_sent\_conn \\ +(-0.013) * verb + 0.162 * noun + (-0.001) * fre \end{aligned} \tag{5}$$

$$\begin{aligned} Y = (-0.094) * fre + 0.08 * L\_sd + 0.043 * HM\_sent\_conn \\ +(-0.031) * sent\_conn + 0.031 * dcr + (-0.03) * verb \\ +0.024 * L\_avg + 0.018 * ttr\_pc + 0.012 * noun \end{aligned} \tag{6}$$

Although, the model does not predict any metrics with zero coefficient, we see from Figure 28, the lowest coefficient values are assigned to the metrics *fre*, *sent_conn*, *noun* and *verb* frequency distributions. The other metrics with lower power of distinguishing between human and non-human (generated) writing are the percentages of noun and verb usage. L_avg or $\mu_{SL}$ and L_sd or $\sigma_{SL}$ are not the top ranked metrics because of high sparsity in sentence lengths in human and machine-written content. Also, while L_sd is positively correlated to the label ($Y$), L_avg has a negative correlation coefficient.

The same metric ranking experiment was carried out for the generated email samples. The interaction of the label $Y$ with the metrics along with the regression coefficients are shown in Equation 6. The regression coefficients of each metric can also be seen in Figure 29. Similar

to the values shown in the story generation ranking experiment, there are no metrics with zero coefficients. The maximum positive correlation is shown by *L_sd* metric along with HM_sent_conn with the second ranked metric with positive correlation. We also observe that DCR metric also has a moderate positive correlation with the label. The readability metric, FRE has the highest negative correlation with the label.

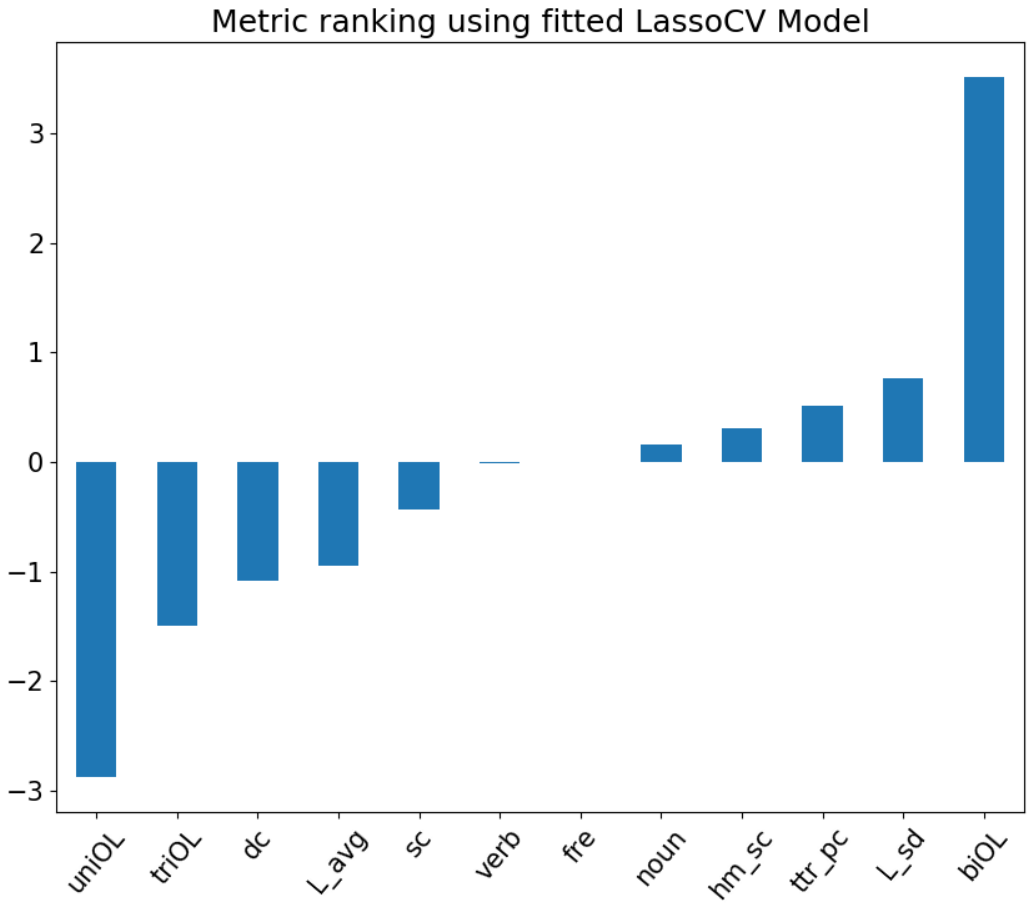



Figure 28: Metric Ranking with Lasso Regularization Method for Stories

### 5.2.2 Analyzing Stories and Emails

The performance of the evaluation metrics on the stories and the emails is shown in this section. We report and compare the best set of metrics from the performance review in the previous section. We

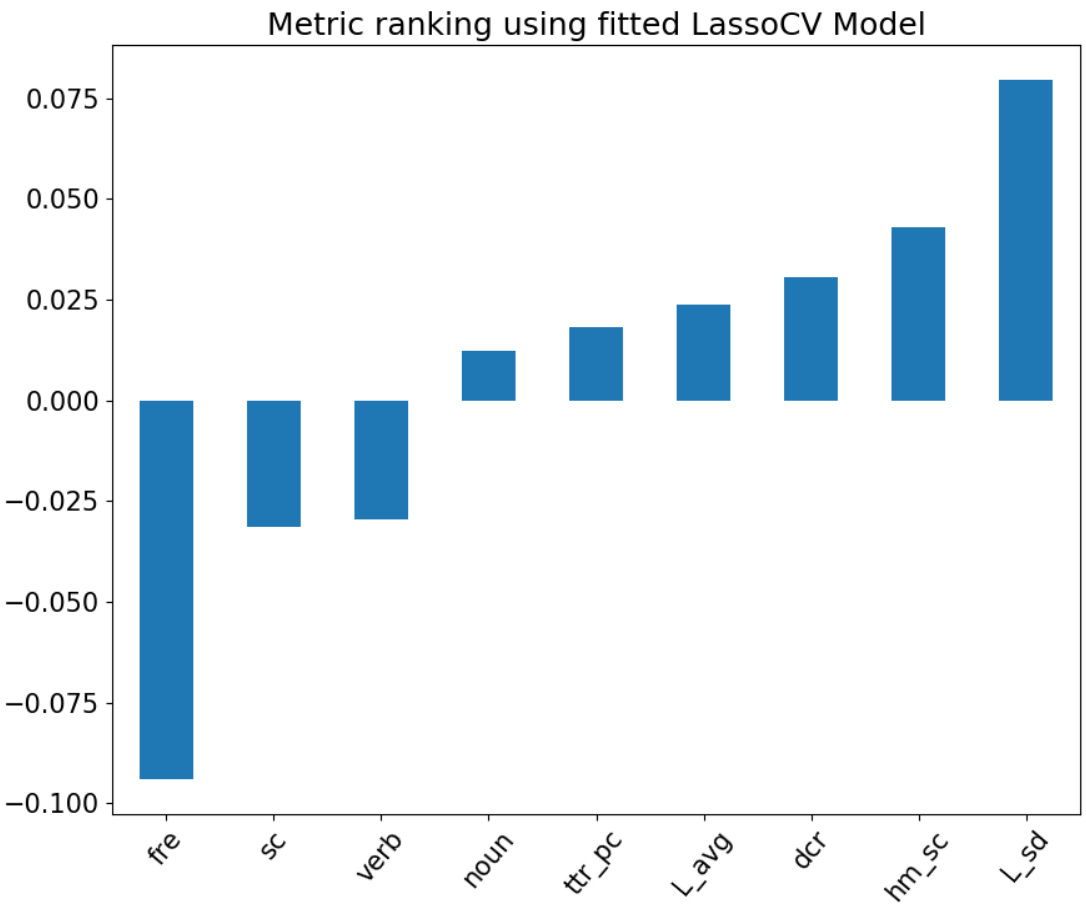



Figure 29: Metric Ranking with Lasso Regularization Method for Emails

analyze the prompt-text overlap, followed by the study of standard deviation of sentence length and type-token ratio. The coherence in the generated text is evaluated using the sentence connectedness metric.

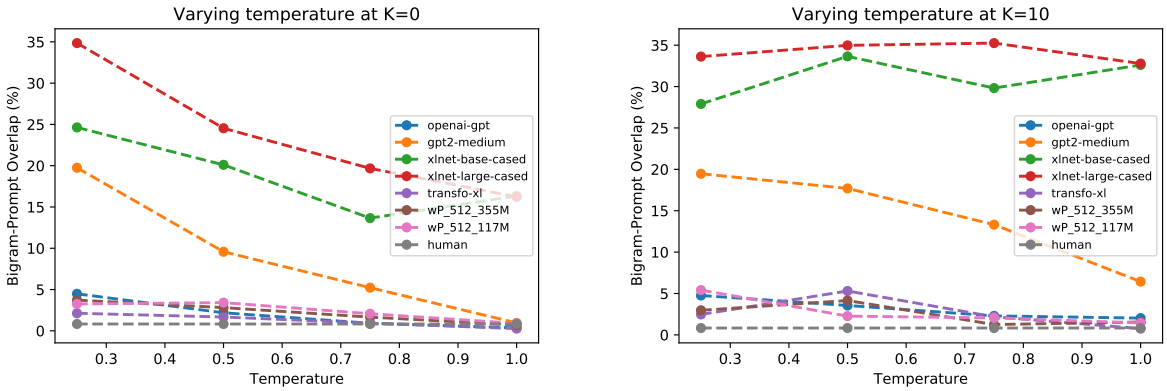
**Story-Prompt Relatedness and Email Readability.** We show the performance of the pre-trained and fine-tuned language models (the top performing systems) on the generated story samples using the *biOL* metric. The metric measures the model's responsiveness to a given prompt. Figure 30 shows the metric scores for the top pre-trained and fine-tuned language models with varying temperature at different values of top-$k$ sampling with respect to human evaluation. The human level of bigram overlap percentage is low and close to 0.5%. A higher overlap percentage

means that the generator is repeating words from the prompt, as seen in the samples from the pre-trained models at lower values of hyperparameters. Among the pre-trained models, the XLNet (XB and XL) and larger GPT2 models (GM and GL) generate samples that are not close to the human references. However, the pre-trained TX and smaller GPT models OG and G2 generate content having minimal overlap (approx. 1%) with the story prompts at temperature 0.75 and $k$ values greater than 150. This is a sign that the models may be generating unique words. More details can be found in the paper [20].

However, for generated emails, text-prompt relatedness measures were either too high or too low, due to the short nature of the provided prompts. We therefore looked at the readability metric score – Dale-Chall Readability (DCR). Figure 31 shows the change in email readability with the variation in temperature $\tau$ for different constant values of $k$. We observed the best set of metric results at sampling $k$ values of 50 and 10, where the scores of the language models are closest to human scores. At higher values of $k$, the metric behavior is slightly unpredictable.

**Syntactic Style and Complexity – Standard deviation of Sentence Length.** Standard deviation of sentence length ($L_{sd}$ or $\sigma_{SL}$) for the generated text in stories and emails are shown in the Figures 32 and 33, respectively by varying the temperature for different constant $k$ values. For the stories, we observe that the $L_{sd}$ in general has a more stable trend for all the models except the XLNet-based models. The trends are closest to the human scores at $\tau = \{0.5, 0.75\}$. The fine-tuned models perform the best with scores closest to human writing. The scores for $L_{sd}$ on the generated emails do not have a stable trend overall, but the fine-tuned language models perform much better with the values being closest to human scores at temperature, $\tau = \{0.5, 0.75\}$. Most models generate samples that score similar to human writing at $k$ values of 50, 250, and 1000, as seen from Figure 33.

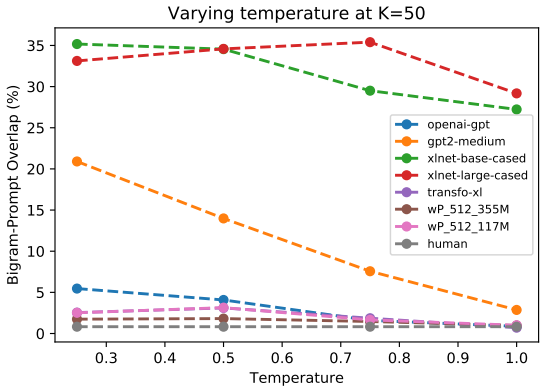
**Syntactic Style and Complexity – Type Token Ratio.** The Type Token Ratio (TTR) measures sentence complexity, lexical richness or variety in vocabulary of the generated content. Studying the TTR values of the generated gives insight into the nature of the generated content based on the class of the generative model. The TTR values of the pre-trained and fine-tuned



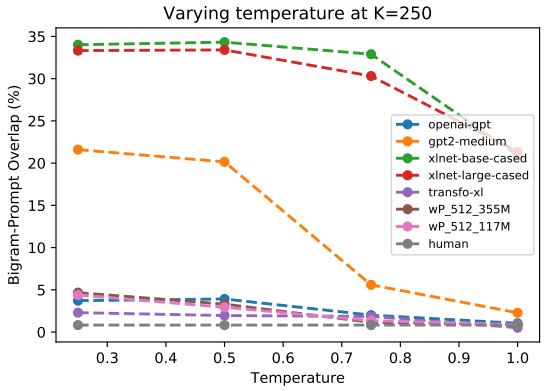

(a)



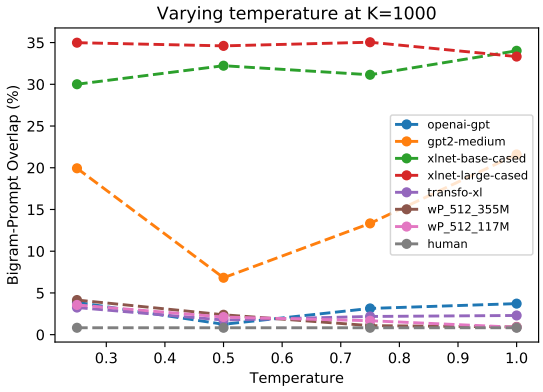

(b)




(c)

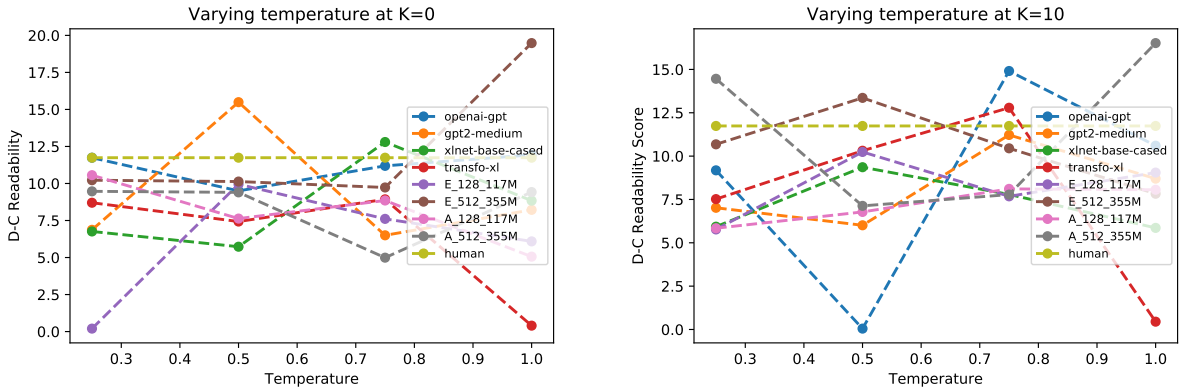



(d)




(e)

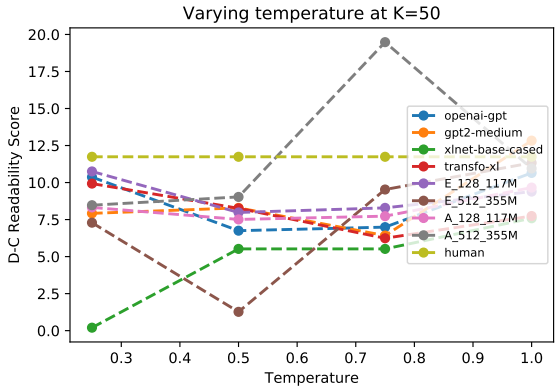
Figure 30: Comparing all models for Bigram-Prompt overlap in stories by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.



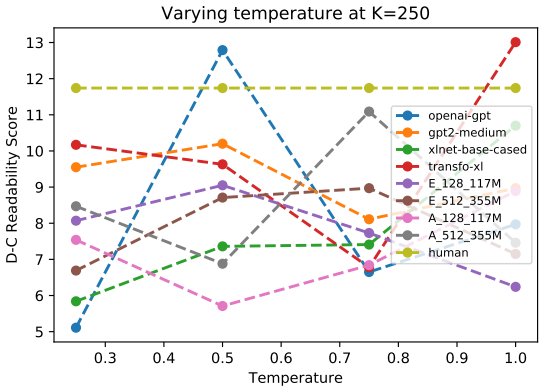

(a)



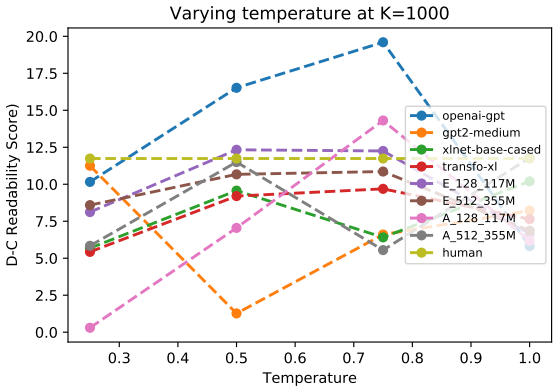

(b)




(c)




(d)




(e)

Figure 31: Comparing all models for Dale-Chall Readability in generated emails by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.

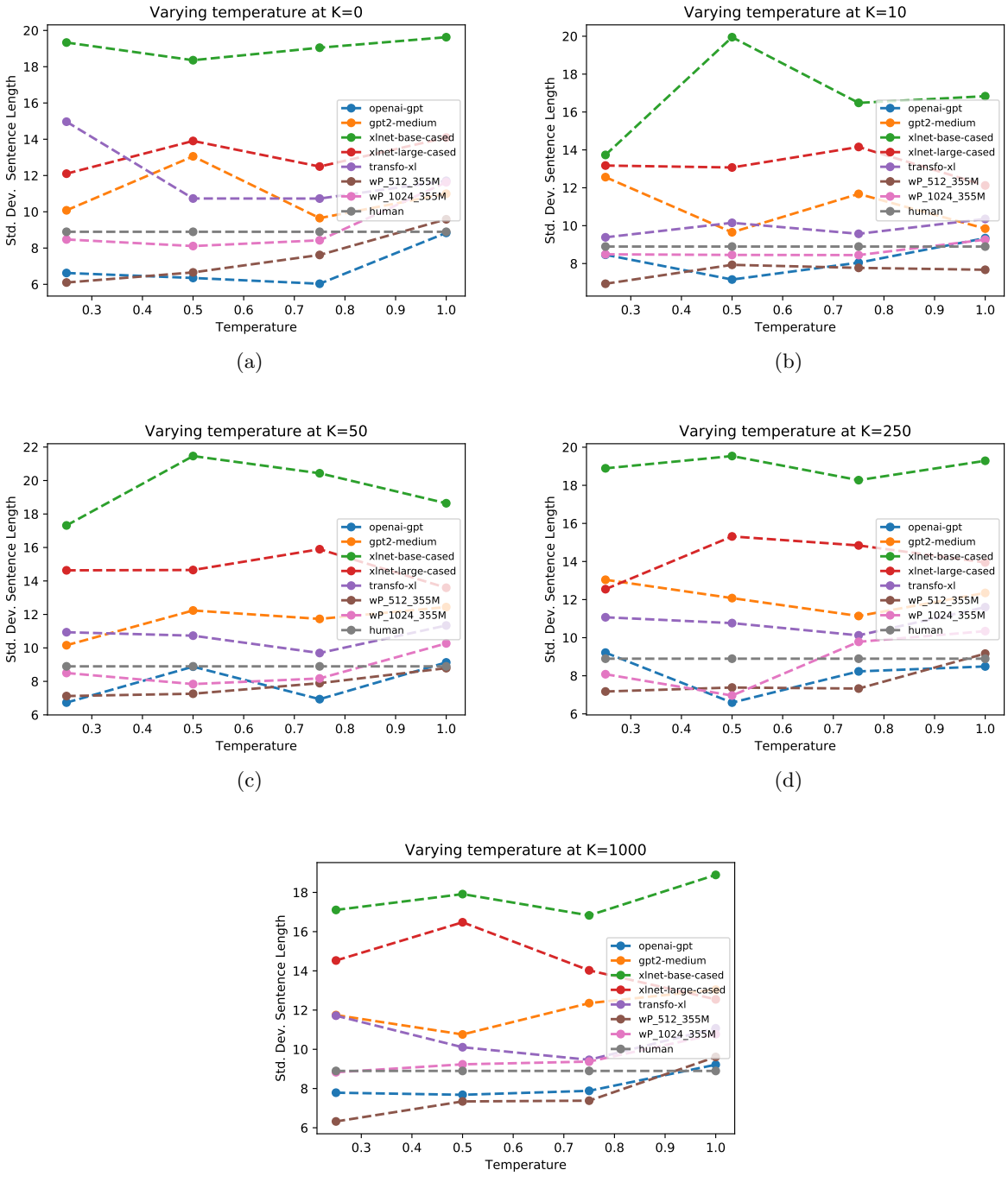
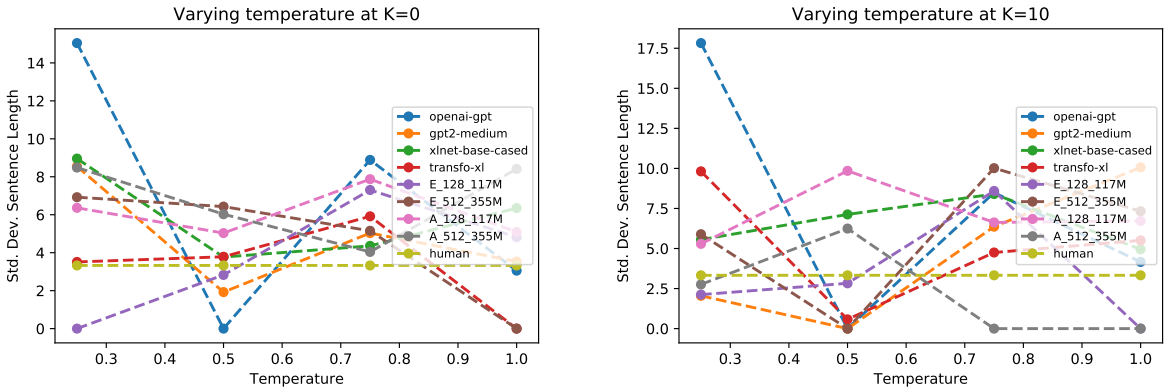


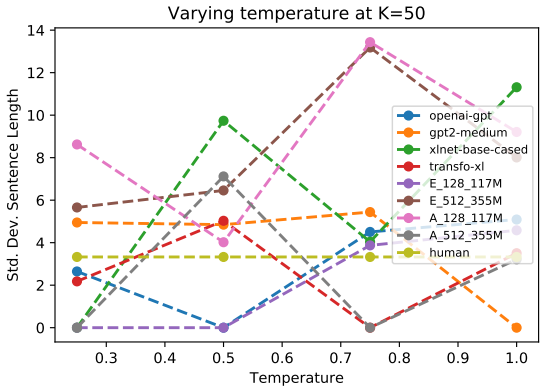

Figure 32: Comparing all models for Standard Deviation of Sentence Length in stories by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.



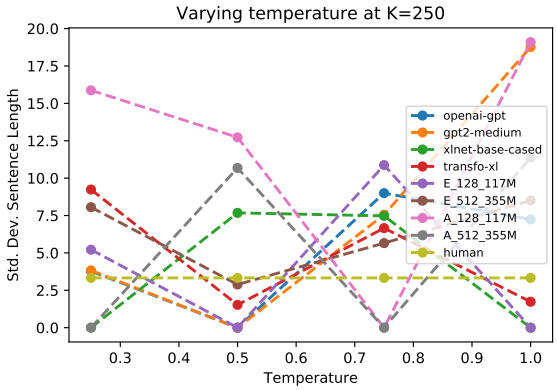

(a)



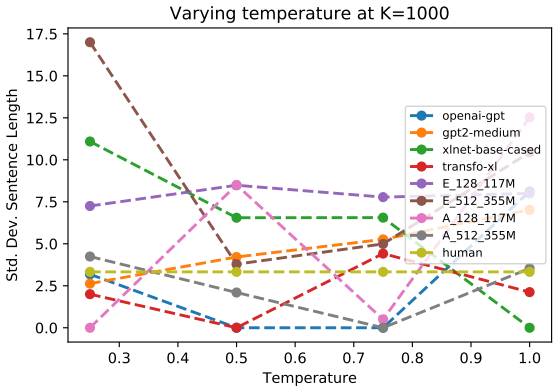

(b)




(c)




(d)



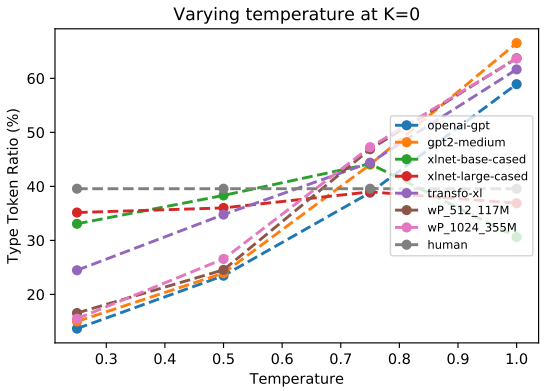

(e)

Figure 33: Comparing all models for Standard Deviation of Sentence Length in emails by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.

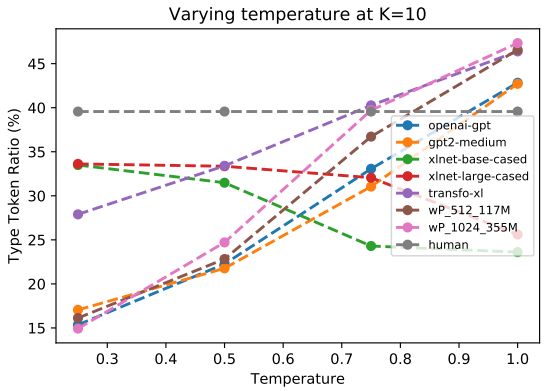
language models for the generated stories are shown in Figure 34 across different constant $k$ values. The figures show that the XLNet model samples have values close to human reference level. The models Transformer-XL, OpenAI-GPT and GPT2-Medium have values comparable to humans at $\tau = 0.75$. The fine-tuned language models do not perform close to the human scores at $\tau = 0.75, 1.0$. The TTR percentage metric for the email samples generated by varying temperature values are shown in Figure 35. Like most of the other metrics shown above, the scores of the TTR percentage vary widely with the model nature and the sampling hyperparameter combinations. We see that like stories, the closest scores are recorded by samples generated at $\tau = 0.5, 0.75$. The fine-tuned models E5-3M get the closest to the human scores at $\tau = 0.75$ for $k = 50$. For higher values of $k$, the scores deviate further away from the human level.

**Harmonic Mean of Sentence Connectedness Statistics.** This sentence-level cohesion measure looks at the harmonic mean between the mean and standard deviation of the connectedness across sentences in the generated content. The connectedness is measured by the angular deviation between the vectorized representations (Sent2Vec[45] embeddings) of each sentence. The performance of the models with respect to this metric is shown in Figure 36 with varying $\tau$ for different constant values of $k$. It shows that the text generated by the fine-tuned and pre-trained language models at higher $\tau$ values score close to human writing. The pre-trained models having the best sentence connectedness scores are GPT2-large and GPT2, along with the w10-3M language models. The harmonic mean of sentence connectedness values, for varying temperature at constant $k$, are shown in Figure 37 for the generated emails. The figure shows that while the human level of the score is very low (approx. 0.05), a large number of the trained language models also score values close to 0.0 – i.e., the variation in the sentence length are very low, thus making the harmonic mean almost negligible. Therefore this metric can be relatively good to determine the author of the textual content with respect to emails. However, at $k = 50$ and $k = 0$, we see the fine-tuned language models E5-3M and A5-3M generating samples that have non-zero HM_sent_conn scores close to human levels.

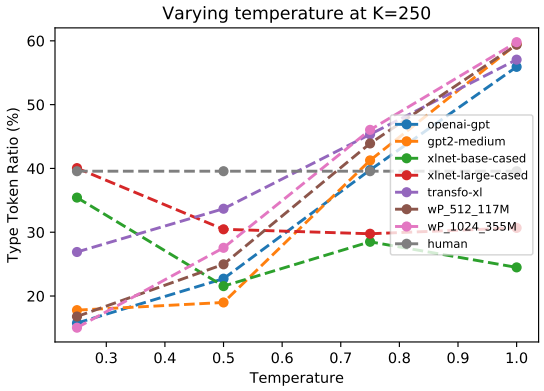
[45] https://github.com/epfml/sent2vec



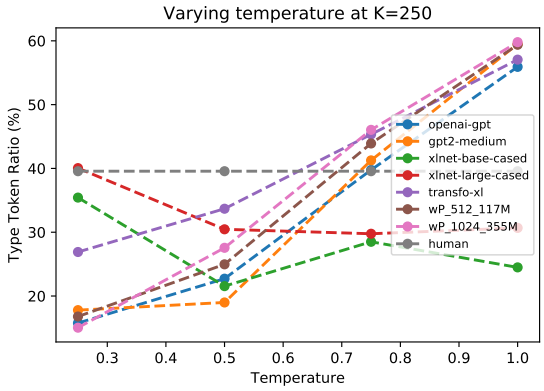

(a)



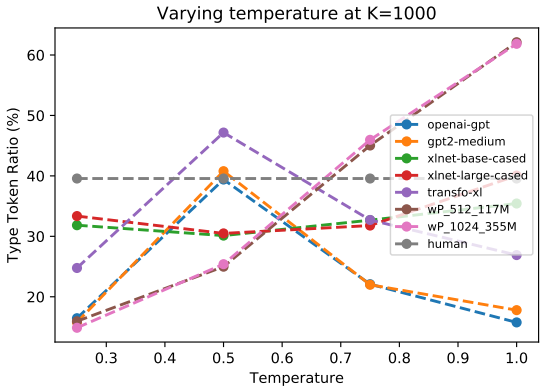

(b)




(c)

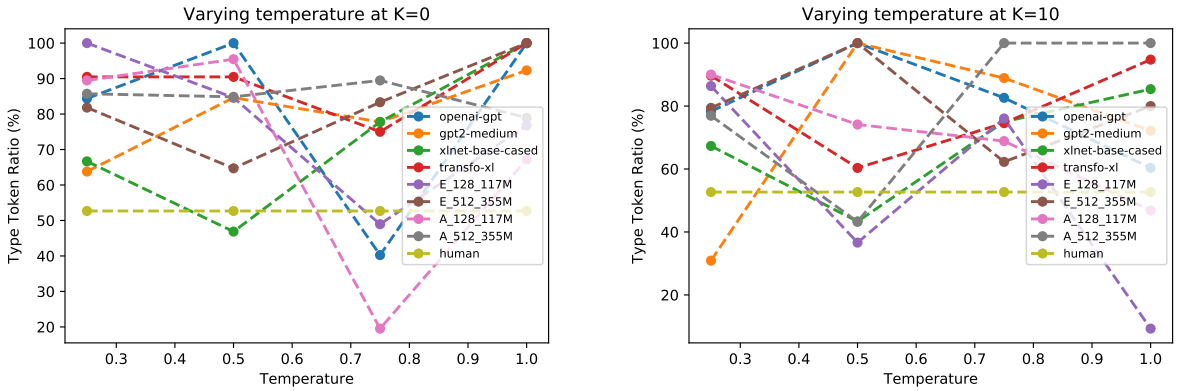



(d)




(e)

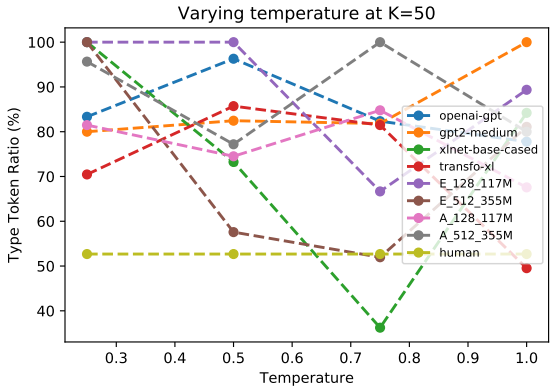
Figure 34: Comparing all models for Type Token Ratio in stories by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.



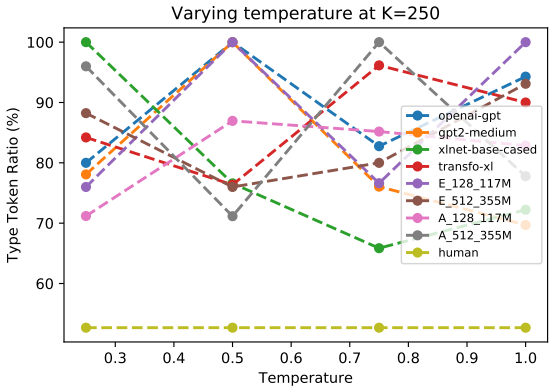

(a)



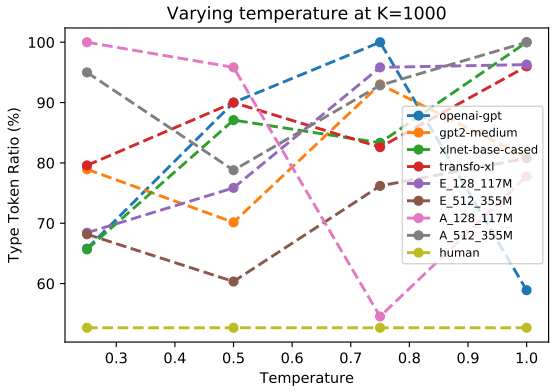

(b)




(c)

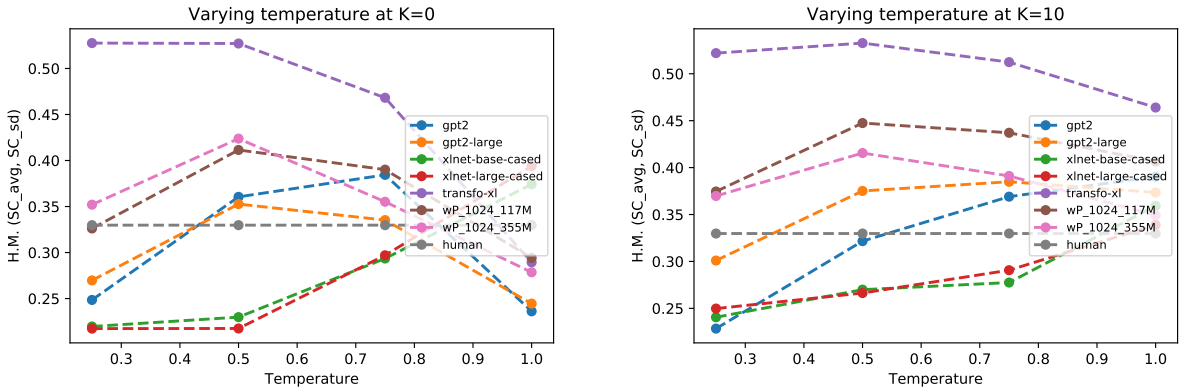



(d)




(e)

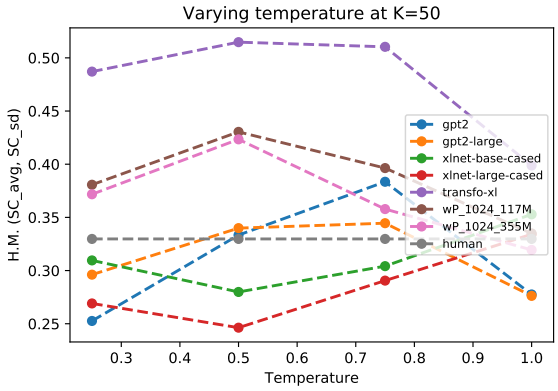
Figure 35: Comparing all models for Type Token Ratio in emails by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.



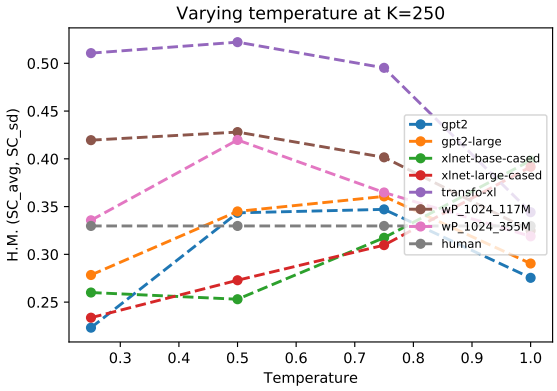

(a)



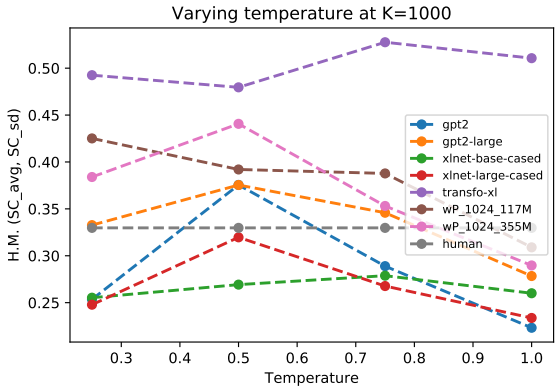

(b)




(c)

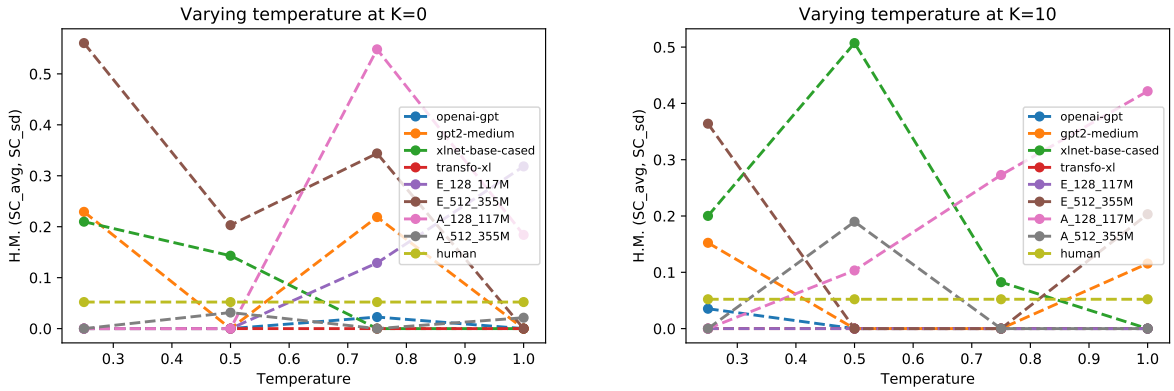



(d)




(e)

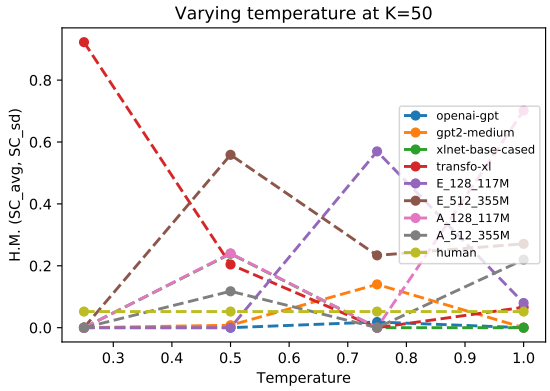
Figure 36: Comparing all models for Harmonic Mean of Sentence Connectedness in stories by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.



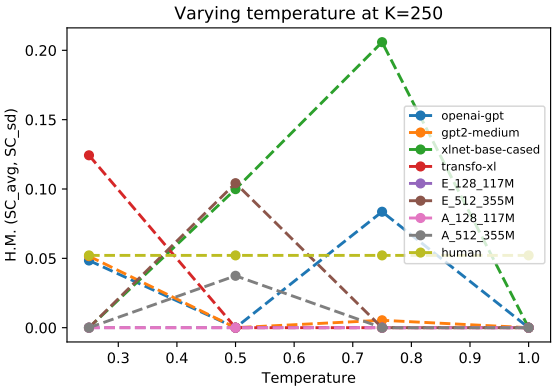

(a)



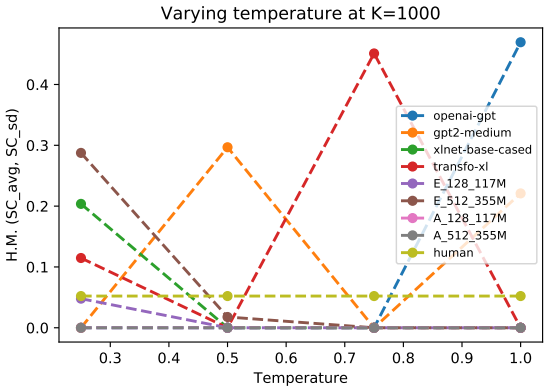

(b)




(c)




(d)




(e)

Figure 37: Comparing all models for Harmonic Mean of Sentence Connectedness in emails by varying temperature ($\tau$) at constant $k$ where (a) $k = 0$. (b) $k = 10$. (c) $k = 50$. (d) $k = 250$. (e) $k = 1000$.

### 5.2.3 Detection Algorithm for Emails

Detecting a synthetic email from human written emails is treated in this study as an anomaly detection problem. We use a OneClass SVM classifier to differentiate legitimate or human authored emails from synthetic or fake or machine generated emails. The detection filter is built using the implementation available as part of the Python ScikitLearn toolkit.[46] We train the model on text-based features, TFIDF count vectors extracted as features from 30,000 Avocado and 30,000 Enron human-written emails. Therefore, the detection model is built on known benign organizational emails and therefore should classify other synthetic machine-generated emails as outliers or anomalies. For testing purposes, we extract TFIDF features from 500 emails generated by each of the fine-tuned models E5-1M and A5-1M as well as the pre-trained language model OG, i.e., a total of 1500 emails to determine system performance, along with 3000 legitimate samples from Enron and Avocado datasets not present in the training set. Table 14 gives the confusion matrix for the generated emails. We considered the human-authored emails as legitimate or Positive (P) class and the synthetic or fake emails as negative (N) class. The precision is 79.5% and the recall is 78.27% for the detection filter. Therefore, lower precision and recall scores indicate that the classifier is not performing well at detecting synthetic emails from legitimate ones.

Table 14: Confusion matrix for Synthetic Email Detection

| | **Actual** | |
|---|---|---|
| **Predict** | P | N |
| P | 2348 | 607 |
| N | 652 | 893 |

## 5.3 Qualitative Analysis

Here, we present the samples generated by the different transformer-based pre-trained and fine-tuned language models for emails and stories.

[46] https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

### 5.3.1 Generated Samples of Stories

For each of the selected test prompts, we randomly select a human-written story from the test set and use the entire 1000 word story for the comparison during the evaluation step. We include representative examples of generated stories from prompts for different language models – both fine-tuned and pre-trained. Using a single prompt, we present some sample examples for different modeling setups in Table 15. We see different language models generating different samples at different combinations of $k$ and $\tau$ in Tables 16, 17 for pre-trained models and Table 18 shows an example of one generated by a fine-tuned one.

Table 15: An example showing a prompt and the stories generated using the **transformer-based** models at different sampling hyperparameters.

| **Prompt** | Whilst restocking a vending machine, a janitor finds something horrific jammed inside. |
|---|---|
| **OpenAI-GPT** $k = 250, t = 0.75$ | he looks around and finds the one thing he is looking for - a bunch of metal bars. he opens them and finds a black metal box. he unscrews the top and pulls out the box. inside is a black metal box. he opens it and finds a length of metal wire. he opens the box and finds a chain, attached to a silver chain [...] |
| **Transformer-XL** $k = 250, t = 0.75$ | The janitor explains that the janitor was a peasant who had been killed by his own brother when he was a baby. He says the butcher had killed his brother, and the butcher had not, and was later brought back to life. Although this story is generally thought by historians to be fabricated, the janitor says the story may have been invented by a man named [...] |
| **W512-117M** $k = 250, t = 0.75$ | the janitor is not the first janitor of this dangerous job. the janitor is one of the janitors that is killed by a bomb. the janitor is too late, a bomb explodes on the janitor's head, killing him . the janitor is the janitor's favorite. the janitor never stops falling, he just fits in like a glove. a janitor dies at the first warning, the janitor is no longer janitor . [...] |

### 5.3.2 Generated Samples of Emails

In this section, we include some email samples generated by the language models. We include examples of emails generated by pre-trained transformer models as well as models trained on BPE-tokenized Avocado and Enron datasets. For each example, we provide the sampling hyper-parameters used for generation. In box 5.1, we present a sample generated by the GPT2-large pre-trained language model. In boxes 5.2 and 5.3, we provide samples generated by the fine-tuned language models – A_128_117M and E_128_117M respectively.

Table 16: An example showing a prompt and a generated story using the **pre-trained gpt2-large** model at $k = 0$ and $t = 0.5$

| | |
|---|---|
| **Prompt** | legends tell of the elder race who utilized strange " technologies "<br>instead of magic . as hordes of brutal killers invade from the north ,<br>a young elf heads off to search for ancient ruins , looking for humans .... |
| **gpt2-large**<br>$k = 0, t = 0.5$ | The first time I saw the movie "The Matrix"<br>I was so blown away by the amazing visual effects that I had to watch it again and again.<br>I was so impressed with the way the movie was shot that<br>I immediately started researching how to make the same effects in real life.<br>I was able to find some amazing resources on the internet and<br>I decided to share them with you. 1. The Matrix Reloaded<br>The movie "The Matrix Reloaded" is a great example<br>of how to make the same effects in real life. It is a film that is very similar<br>to the original "The Matrix" but has a few differences.<br>The main difference is that the characters in the movie<br>are not actually in the Matrix |

Table 17: An example showing a prompt and a generated story using the **pre-trained gpt2-medium** model at $k = 250$ and $t = 1.0$

| | |
|---|---|
| **Prompt** | Your personal life is a complete wreck, but you never cry about it,<br>thanks to this tear-free shampoo. |
| **gpt2-medium**<br>$k = 250, t = 1.0$ | That'll keep you warm and happy from start to finish.<br>I'm really enjoying my love with this product, and its amazing for<br>keeping my makeup-blonde and black lines out of my eyes - with every wash.<br>It also doesn't smell and feels great on my skin :).<br>LOVE IT. LOVE IT.<br>SUGGESTIONS:<br>1. Please ask your natural ophthalmologist for advice if you're<br>having your water color problem.<br>2. Avoid using olive oil in your water-colored lotions when your<br>eyes are feeling oily - it's greasy!<br>For more info on Color Enhancers/Watercolors visit www.amazon.com<br>Read More... |

**Box 5.1: Samples generated by pre-trained GPT2-Large language model**

**1. Conditional Sampling at $\tau$ = 0.75 and $k$=50:**
*Email Subject: Attachment*
accepted yesterday re TIm stack + OC user skill is prototype but client failed introl cancelled # 6093 should update for 4 seconds pwl insbh has attached please communicate status comments are inddependent.

Table 18: An example showing a prompt and a generated story using the **fine-tuned wP_512_355M** model at $k = 150$ and $t = 1.0$

| **Prompt** | Tikibara lagoon |
|---|---|
| **wP_512_355M** $k = 150, t = 1.0$ | As i approached the wave, the soft-pearl glow of the night reflected the shape of each life as i arrived close to the water's edge.<br>my ship docked at the head and the ship's holographic display began to turn blue and all traces of time faded away.<br>from the inside of the helmet, i knew my mission was accomplished,<br>i had just accomplished what i had set out to accomplish in search of a new kind of energy for humanity. walking toward the crew i realized,<br>oh, what had happened to me. i lost all meaning in the form of time.<br>everything started happening again i experienced life again for the first time.<br>i understood this all had to be done. |

**Box 5.2: Samples generated by Avocado-128**

**1. Conditional Sampling at $\tau$ = 0.75 and $k$=50:**
*Email Subject: 3.5 integration plan*
accepted yesterday re TIm stack + OC user skill is prototype but client failed introl cancelled # 6093 should update for 4 seconds pwl insbh has attached please communicate status comments are inddependent.
**2. Sampling at $\tau$ = 1.0 and $k$=500:**
Yes, the meeting will start at 3.00pm.

**Box 5.3: Samples generated by Enron-128**

**1. Conditional Sampling at $\tau$ = 0.75 and $k$=50:**
*Email Subject: dow jones distribution agreement*
needs edits before posting - John picked Tues 2 m ago ... do mord unparks this and talk to u see
**2. Unconditional Sampling at $\tau$ = 1.0 and $k$=500:**
I just spoke to *ent* with regards to the details on hedging scot for energy we are thinking of taking off next week.

# 6 Text Generation with HiGen

We describe in detail the proposed hierarchical generative architecture or **HiGen** in Section 6.1. Section 6.2 explains the experimental setup for training and evaluating the proposed architecture on two different genres of datasets – stories and emails. A quantitative analysis of the automated metrics is provided along with inter-metric correlation study and ranking in Section 6.4. Finally, Section 6.5 provide a qualitative analysis of the generated samples for both genres – stories from prompts and emails.

## 6.1 Proposed Sentence-Level Architecture

Using a 'simple' pre-transformer recurrent neural network or LSTM units for content generation has led to issues like incoherent text sequences, digression from topic, etc. [34]. Controlling these properties of the generated text automatically becomes a challenging task. However, with the emergence of transformer architecture [95] attention modeling in deep neural architectures helps learn better representations of the text on which these architectures are trained. The transformer-based massive language models like OpenAI's GPT-2 [72, 73] and Google's Transformer-XL and XLNet [19, 107] have shown exceptional performance in the realm of language generation tasks [66, 2]. But these models also suffer from conditioning issues: the models being trained on large amounts of textual content often run off-topic or cannot condition properly on the given starting prompt or seed [79]. Hence, to combat these issues of coherence and prompt-conditioning, we propose an hierarchical architecture, which leverages the above proposed language models for text generation. Additionally, we add a sentence-embedding level predictive language model that selects the best generated candidate sentences from the generative model. The architecture therefore models the generated text by controlling coherency and topic consistency at a sentence level.

In this section, we describe the different sections of the proposed deep neural architecture. Figure 38 shows the architecture of the neural learning algorithm that we propose and evaluate. The architecture is given a collection of documents as input. Depending on the task at hand, in this

research, we consider two major types of data – stories from the WritingPrompts dataset [27] and human-authored emails from the organizational email corpora, Enron [26] and Avocado [23]. To start off the sample generation, the architecture is provided with a set of starting seed sentences. As seen from the figure, the architecture is hierarchical in nature consisting of two models – a sentence candidate generation language model and a predictive sentence selection language model. The sentence selection model ranks the generated candidate sentence and selects the best novel generated candidate based on a similarity metric score. The generation and selection of the textual content continues iteratively using a feedback loop until an end criterion is reached.

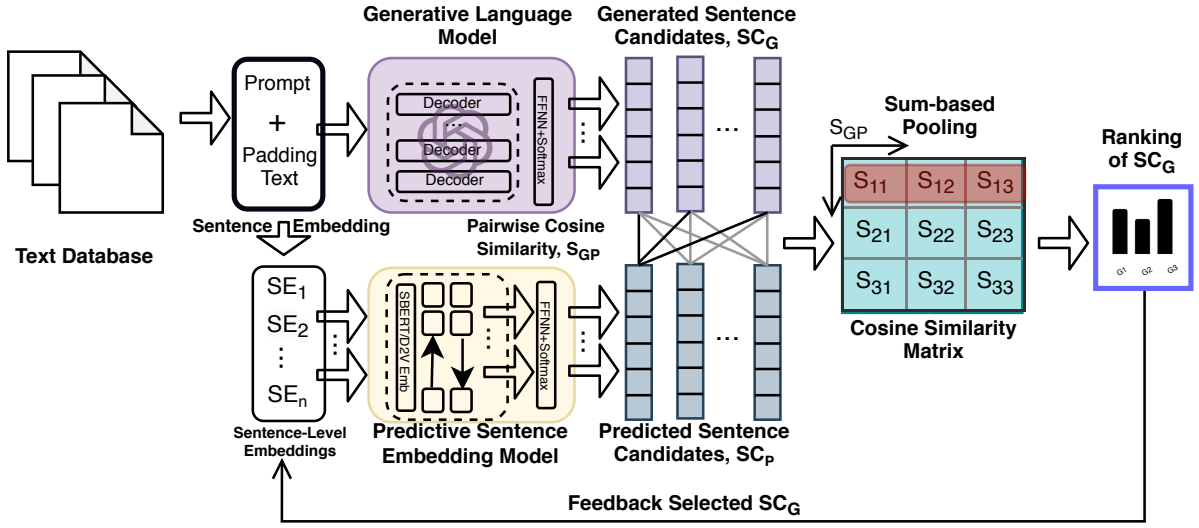



Figure 38: Sentence based Generative Architecture

We discuss the proposed architecture and the purpose of the models in detail in the following sections. The architecture is divided into three major parts: *(a) Sentence generation architecture* – the fine-tuned language model that generates a set of N novel sentence phrases, that we call *generated sentence candidates, $SC_g$*; *(b) Sentence embedding prediction architecture* – a sentence embedding-based model trained to predict the best set of sentence candidates from the predictive model in the form of sentence vectors, *predicted sentence candidates. $SC_p$*; and finally, *(c) Selection module* – this module calculates the pair-wise similarity scores between the candidate generated and predicted sentence vectors - $SC_{gi}$ and $SC_{pi}$ respectively, where $i \in N$, i.e., the total number

of candidates. Based on the similarity scores, the best sentence candidates are finally chosen from $SC_g$ as the best generated sentence.

The main aim is to ensure a more uniform and coherent generative scheme by controlling the textual content generation at the sentence-level and using a trained sentence-embedding based language model as the guide to select the best generated candidate. The guided feedback step is used to provide additional contextual information to the predictive sentence selection model.

### 6.1.1 Sentence Candidate Generation Model

The generative language model generates candidate sentences given a starting seed or a prompt. This module consists of either a word-based RNN architecture trained on textual content (stories) or a generative transformer language model (pre-trained or fine-tuned) as the generative model. The word-based language model generates text phrases of length $N$ words (here, $N = 10$). We provide a starting seed sequence of length 20 words – the prompt is the writing prompt for story generation along with padding text taken from the human-authored story for the given prompt. For the email generation task, the email subject acts as the prompt.

At the start of generation, the generative language model takes the starting prompt and generates the $N$ best suitable and novel candidates given the starting prompt ($SC_{g1}$, $SC_{g2}$, ..., $SC_{gN}$). In this implementation, for the word-based RNN model we vary the temperature hyperparameter to generate 5 sentence candidates in an iterative fashion. For the generative transformer models, we generate six probable sentence candidates by using a combination of temperature ($\tau$) and top-$k$ sampling parameters. The range of values for these parameters are: $\tau = \{0.75, 1.0\}$ and $k = \{150, 250, 500\}$, chosen through a metric based study as outlined in [20]. These generated sentence candidates $SC_g$ are then passed on to the next module for ranking and comparison through similarity metrics.

### 6.1.2 Sentence Vector Prediction Model

The starting seed is provided to the vector prediction model. This module is a predictive language model built by training on sentence embeddings extracted from the datasets (story/emails). In this architecture, we test with two types of sentence embedding methods - Sentence-BERT [74] or S-BERT and Doc2Vec [57] or D2V. For the S-BERT embedding model, we extract the sentence embeddings by using a pre-trained base BERT uncased language model trained on natural language inference dataset. These S-BERT sentence vectors are 768 dimensions in length. The D2V model for encoding the document vectors is built by training on the task dataset for 50 epochs building 500-dimensional vectors.

Finally, the extracted sentence embeddings are used to train a Bi-LSTM based language model [38] with sentence vector sequences of length 5 (i.e., five sentence vectors in each sequence). The predictive architecture is built using 256 units (i.e., total 512 Bi-LSTM units) with an initial learning rate of $10^{-2}$ decaying with a rate of $10^{-4}$. The model was trained for a total of 100 epochs for a total of 7 hours. The optimizer used was *Adam* and the loss function *logcosh* and the accuracy metric used was *categorical accuracy*. The entire system was built in Python Keras (Version 2.3.0) library with Tensorflow 1.13.1 as the backend. For the same given seed prompt, the trained embedding-based model predicts the next $N$ candidate sentences, ($SC_{p1}$, $SC_{p2}$, ..., $SC_{pN}$).

### 6.1.3 Selection and Ranking

Using the sentence candidate generation module, the system generated $N$ sentence candidates, $SC_{gi}$ where $i \in \{1, N\}$ for the given initial starting prompt. For the same prompt, the predictive sentence embedding-based model chose $M$ sentence vector candidates - $SC_{pj}$ where $j \in \{1, M\}$. The $SC_{gi}$ candidates were converted to their vector representations using the trained embedding model used to generate $SC_{pj}$, thus we can call these vectorized candidates - $SC_{vgi}$. We used a Python based cosine similarity metric from Scikit-Learn to calculate the similarity between the $SC_{pj}$ and the $SC_{vgi}$ candidates. For a given generated candidate $SC_{gi}$, we calculated the cumulative similarity score with the predicted $SC_{pj}$ candidates. The $SC_{gi}$ with the highest similarity values was chosen

as the output of the system.

### 6.1.4 Sampling and Feedback

The final step consisted of ranking the chosen candidate sentences and then providing the best selected candidate as a feedback to the models to act as a seed sentence and start off the next round of generation. The $SC_{gi}$ with the highest similarity values was chosen as the output of the system and sent back to the start of the architecture to act as the prompt or the priming sequence for the second round of generation. The generation continued in an iterative fashion until an end criteria was reached. This can be specified in the form of number of complete sentences to be generated as part of the content or an end token like $| < endoftext > |$. We used a length of 5 sentences as a stopping condition.

The main aim is to ensure a more uniform and coherent generative scheme by controlling the textual content generation at the sentence-level and using a trained sentence-embedding based language model as the guide to select the best generated candidate. The guided feedback step is used to provide additional contextual information to the predictive sentence selection model.

## 6.2 Experimental Setup

We designed the hierarchical generative architecture to have two language models – sentence generation model and sentence selection model – acting in conjunction with each other. The output from each of these models was compared in a final ranking and selection module to select the best sentence candidates. In this section, the experimental setup has been discussed, depending on the type of the data. Given a starting seed, the **top generative model** was used to generate the most probable sentence candidates ($SC_g$) to follow the given seed sentence at the first level of iteration. We tested the following language models for generation – pre-trained models (OG and TX) and fine-tuned models (w5-1M, w5-3M and w10-3M for stories and E5-3M and A5-3M for emails). The **second part of the architecture** is the predictive model built on the sentence-embedding

based vectors. We used two sentence-level embeddings to build the vector prediction model - 768-dimensional Sentence BERT or SBERT [74] and 500-dimensional Doc2Vec or D2V [57]. We explain the details of the sentence-level embeddings used in Section 3.3.3 along with the Bi-LSTM based architecture used as the predictive model to generate the $SC_p$ vectors. Below are the two sentence vector prediction scenarios:

- **For the SBERT-based model:** We encoded the training dataset using the pre-trained SBERT sentence embedding model – 'bert-base-nli-stsb-mean-tokens'[47] to predict the most probable 768-dimensional sentence vector given an input SBERT encoded sentence vector.
- **For the D2V-based model:** The Doc2Vec embeddings for encoding were trained on the provided wP-512 dataset using the Python based implementation. The embedding model is used to encode the training data into 500-dimensional D2V vectors which are used to train the predictive model using the Bi-LSTM based network.

The final stage is the **selection of the best candidate.** As mentioned before, we used the cosine similarity metric to calculate the similarity between the sentence embedding based vector of the $SC_g$ and the predicted vector $SC_p$. We used the Scikit-Learn's cosine similarity metric to calculate the similarity between the sentence vectors. The $SC_g$ with the highest similarity to $SC_p$ was selected and fed back to the network to begin the next round of generation.

### 6.2.1 Generating Stories

We describe how the architecture has been used to generate samples of stories from the WRITINGPROMPTS dataset. We discuss the training steps followed by the sample generation. Given a story prompt as a starting seed, the top generative model was made to generate $SC_g$ candidates based on the best sampling hyperparameter combinations of temperature ($\tau$) and top-$k$. The values chosen are $\tau = \{0.75, 1.0\}$ and $k = \{250, 500\}$. Thus, at the first iteration, the model generated 4 $SC_g$ for the given story prompt. The model for candidate sentence vector prediction was built on two

[47] https://www.sbert.net/docs/pretrained_models.html

types of sentence-based embeddings – S-BERT and D2V. The predictive model for S-BERT was built by training the Bi-LSTM based architecture with sentence-level S-BERT encodings on the story generation datasets wP-256 and wP-512. Similar to the S-BERT based model the D2V-based architecture was also trained on the BPE tokenized story datasets for a total of 500 epochs. Finally, the vector representations of the list of generated sentence candidates ($SC_g$) were compared with the predicted vector $SC_p$ to select the best generated sentence through cosine similarity. The $SC_g$ with the highest similarity to $SC_p$ was selected and fed back to the network to begin the next round of generation.

### 6.2.2 Generating Emails

Similar to the setup followed for story generation, we describe the steps for email generation using the hierarchical architecture, in this section. In this case an email subject was given as a starting seed, the generative model was used to generate the most probable sentence candidates ($SC_g$) that follows the seed prompt. As done previously, we generated candidates using samples from pre-trained models (OG and TX) and fine-tuned models (E5-3M and A5-3M). Based on prior analysis, the sampling hyperparameters chosen were: $\tau = 0.75, 1.0$ and $k = 50, 250, 500$. Thus, for every iteration, the generative model generated 6 $SC_g$ for the given prompt. We chose to generate more candidates due to the sparse nature of the email text. Two types of embeddings to build the vector representations - S-BERT - [74] and D2V [57] - based embeddings from the datasets Enron-512 and Avocado-512 (combined) were used to train a Bi-LSTM based architecture to build the sentence-embedding based predictive model to predict the most probable $SC_p$ vector given a starting seed. Finally, the best generative candidate was selected by calculating the cosine similarity between the vectors of $SC_g$ and $SC_p$. The selected candidate was sent back to the model to restart the generation.

## 6.3 Generation of Samples

Using the proposed architecture, we built the language models depending on the different combinations of generative model and sentence embedding models. Based on prior analysis with generative language models [20], we selected the top two pre-trained language models – OpenAI-GPT and Transformer-XL (OG and TX) and the top fine-tuned language models (w5-1M, w5-3M, w10-3M for story generation and E5-3M, A5-3M for email generation) for our generation of sentence candidates. For our sentence embeddings, we used S-BERT and D2V based embeddings as described earlier. Using different combinations of pre-trained and fine-tuned language models in the proposed architecture, we evaluated the performance of 10 language model combinations for the task of story generation and 8 language models for the task of generating emails. In each case, we generated a total of 100 samples.

## 6.4 Quantitative Analysis

The results of the evaluation metrics on the stories and the emails are highlighted in this section. First, we reviewed the correlation among the evaluation metrics that have been discussed in Section 3.5 and ranked their performance based on their scores on human authored and machine generated text. We selected the top performing set of metrics and observed how their performance changes based on the type of model used and the author (human or machine) of the text generated.

### 6.4.1 Metric Correlation and Ranking

The section discusses automated evaluation metrics mainly on two aspects – the inter-metric correlation and metric performance ranking.

**Metric Correlation.** We studied how the results of the evaluation metrics are correlated among themselves. The metric correlation on the stories is shown in the correlation based heatmap in Figure 39. As shown in prior analysis, this figure also shows that there exists a strong correlation among the overlap based metrics (uni-, bi- and tri-gram overlap) and between the readability metric

DCR and type-token ratio. Also, there exists a moderate positive correlation between the length-based stylistic metrics (mean and std. dev. of sentence length) and $n-$gram based overlap.

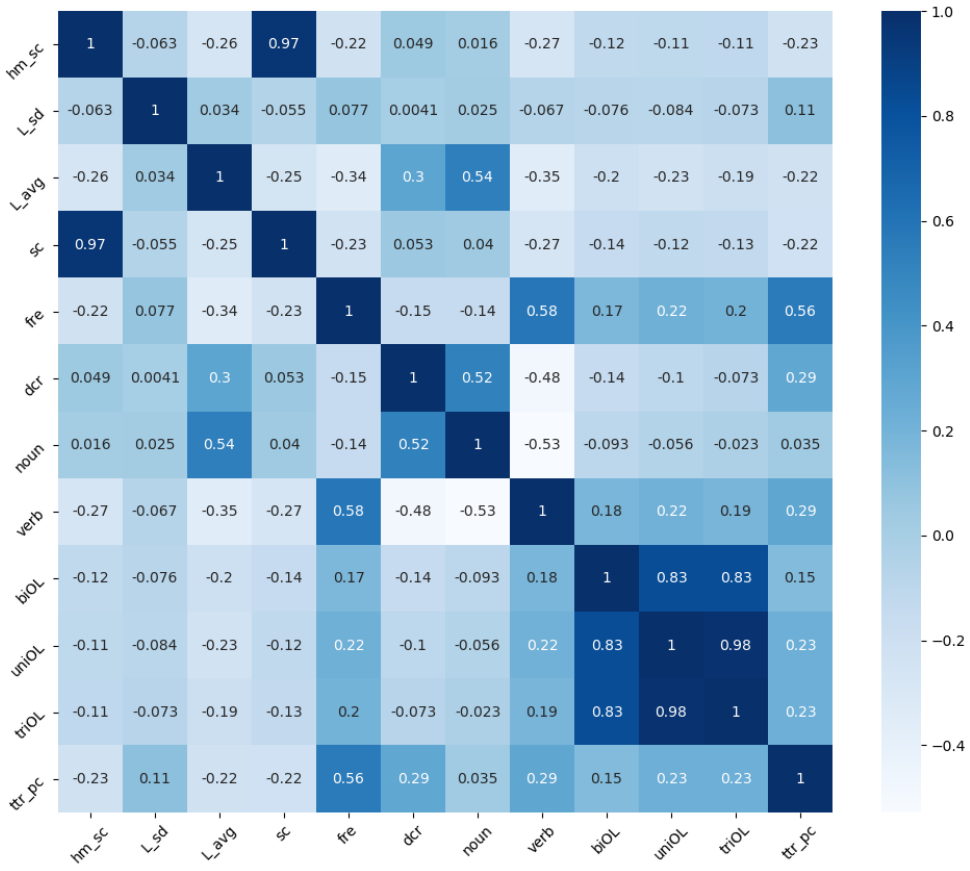



Figure 39: Heatmap showing Pairwise Metric Correlation for Stories generated by Hierarchical Architecture

Figure 40 shows correlation for the metric scores on the generated emails. There exists low correlation among the metrics – this can be attributed to the short text nature of the emails as well high sparsity in the email content. However, there exists a moderate correlation among the POS-tag based metrics (noun and verb) and the TTR percentage values.

**Metric Ranking.** We use LassoCV as the metric performance ranking algorithm to evaluate the ability of a metric to distinguish between human-authored and machine-generated content.

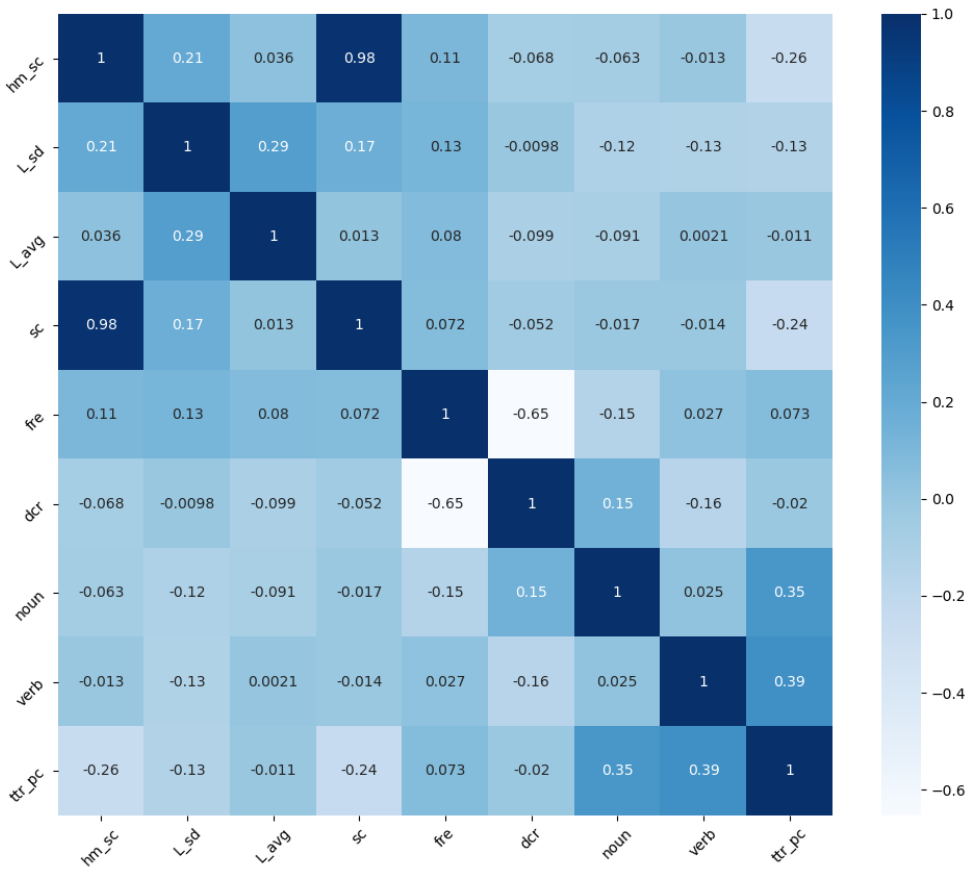



Figure 40: Heatmap showing Pairwise Metric Correlation for Emails generated by Hierarchical Architecture

Depending on the genre of the text, the metrics' performance-based rankings are shown in Figures 41 and 42 for stories and emails, respectively. Equation 7 shows the regression coefficients for the regression analysis on evaluation of stories generated by the proposed architecture. Here, $Y$ refers to the *'label'* or the author of the generated content – machine or humans. The same is shown for emails by Equation 8. The regression analysis follows the steps as described in Section 5.2. For stories, the regression coefficient values are considerably higher than that observed for email samples. We see that the overlap-based metrics are higher ranked along with the metrics like $HM\_sent\_conn$ and POS tag frequencies in Figure 41. Among, the coefficients for emails shown

in Figure 42, *sent_conn* is the highest ranked followed by DCR readability metric. The average sentence length has a high coefficient, although negatively related to the 'label.' For the metric ranking analysis performed using the LassoCV algorithm, we observe that for the stories, the best model score (coefficient of determination, $R^2$) is 76.51% for the stories and the adjusted $R^2$ is 73.6%. For the regression models built for emails, the best $R^2$ is 5.79% and adjusted $R^2$ was 1.49%.

$$\begin{aligned} Y = 0.585 * uniOL + (-0.464) * triOL + (-0.262) * fre \\ +0.245 * verb + 0.241 * HM\_sent\_conn + (-0.172) * dcr \\ +(-0.171) * sent\_conn + (-0.159) * L\_avg + 0.147 * noun \\ +(-0.131) * biOL + 0.088 * ttr\_pc + 0.024 * L\_sd \end{aligned} \tag{7}$$

$$\begin{aligned} Y = 0.089 * sent\_conn + 0.085 * dcr + 0.076 * fre \\ +(-0.067) * HM\_sent\_conn + 0.036 * noun + (-0.034) * verb \\ +0.03 * ttr\_pc + (-0.026) * L\_avg + -0.003 * L\_sd \end{aligned} \tag{8}$$

### 6.4.2 Analyzing Stories and Emails

We observe the performance of the best set of evaluation metrics on the samples generated by the proposed architecture. In this analysis, we compared each metric based on their performance on the genre of the data. To compare among the different models of the proposed architecture for each metric, we review the statistical properties of the scores of each model on the text generated through boxplots. We calculated the statistical significance between the mean of the models' scores and that of the human written content for each metric. We discarded the outliers in our analysis. Based on our prior evaluation, we report the following quantitative metrics – prompt-based conditioning (for stories), stylistic measures of complexity, readability (for emails) and harmonic mean of sentence connectedness.

**Story-Prompt Relatedness and Email Readability.** We see from prior analysis that $n-$gram based overlap measures are highly correlated among themselves with a change in $n$.

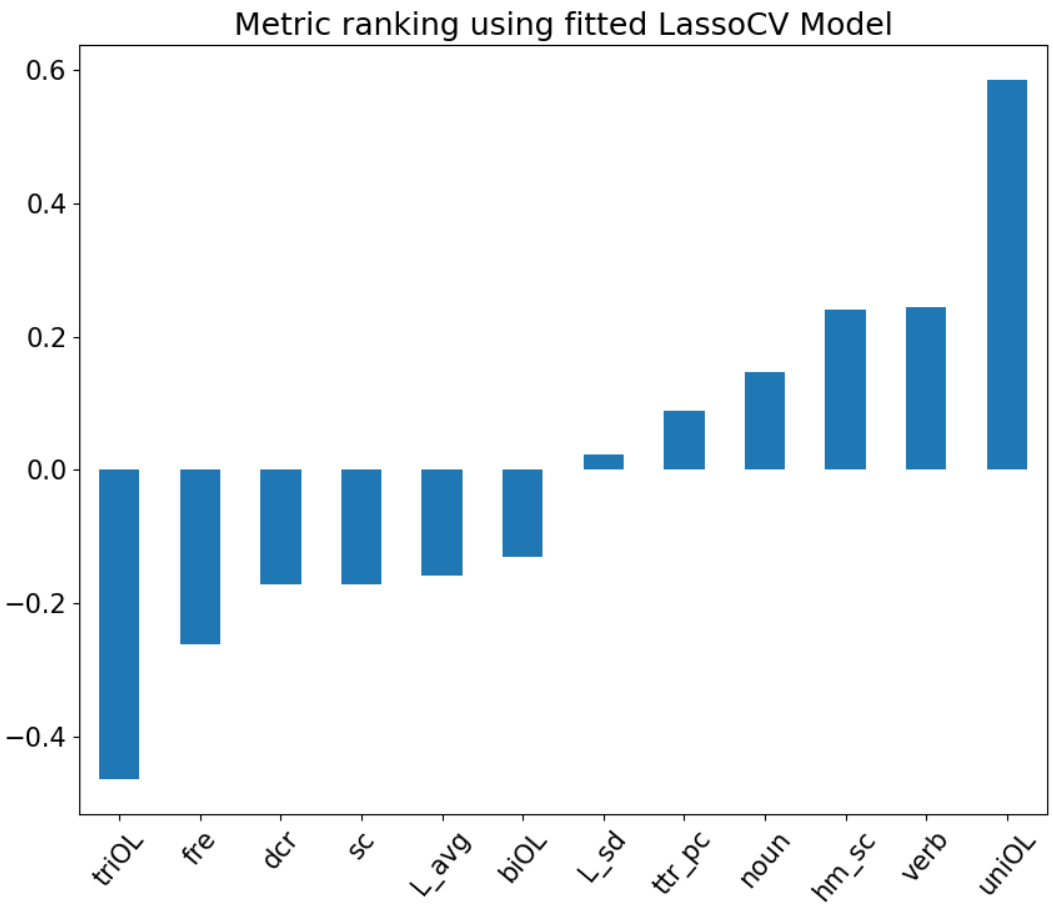



Figure 41: Metric Ranking for Stories with Lasso Regularization Method

To compare with prior quantitative analyses of generated stories, the bi-gram overlap percentage ($biOL$) between the stories and the writing prompt is reported here. We present the evaluation of the hierarchical generative models using boxplots to show the change in the model performance with respect to change in the model nature as shown in Figure 43. The human level of $biOL$ percentage is low and close to 0.5%. For the different generation-selection setup in the proposed HiGen, we studied the statistical properties of $biOL$ of the textual content generated in each setting. We see that the mean $biOL$ percentage scores is statistically significantly higher for all the models with the Transformer-XL models performing the closest to the human scores.

**Syntactic Style and Complexity – Sentence Length.** Mean and standard deviation

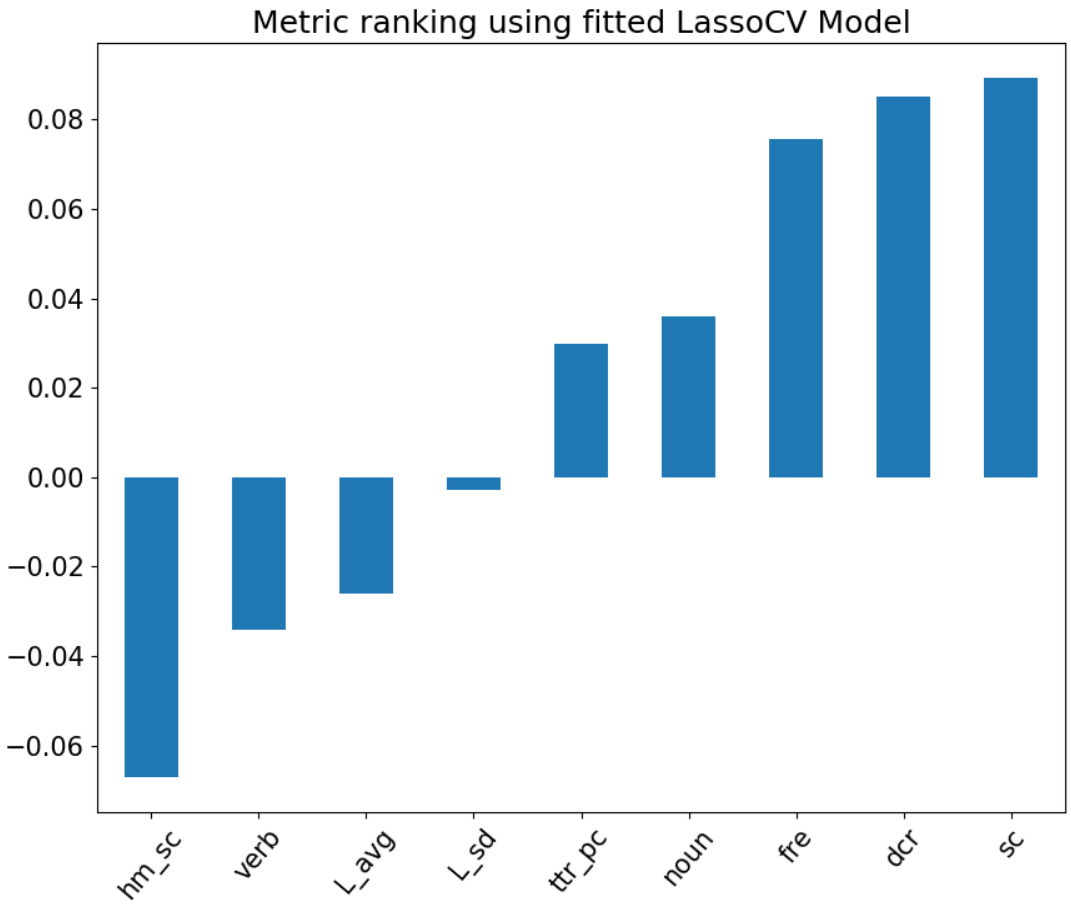



Figure 42: Metric Ranking for Emails with Lasso Regularization Method

of sentence length provide a good overview of the stylistic properties of the generated content. We analyzed the standard deviation of sentence length ($L_{sd}$) in the content generated by the different language models. The boxplots in Figures 45 and 46 give statistical properties of the $L_{sd}$ values for the story and email samples generated by the different models of the HiGen architecture respectively. For the generated stories, we see the mean of $L_{sd}$ values of the fine-tuned language models (w5-1M_D2V and w10-3M_SBERT) closest to the human scores but still not statistically significantly similar. Figure 46 shows the scores across the HiGen models for the generated emails. We observe that the fine-tuned model variations perform the best with scores closest to human writing. However, the scores are not statistically significantly similar to human writing.

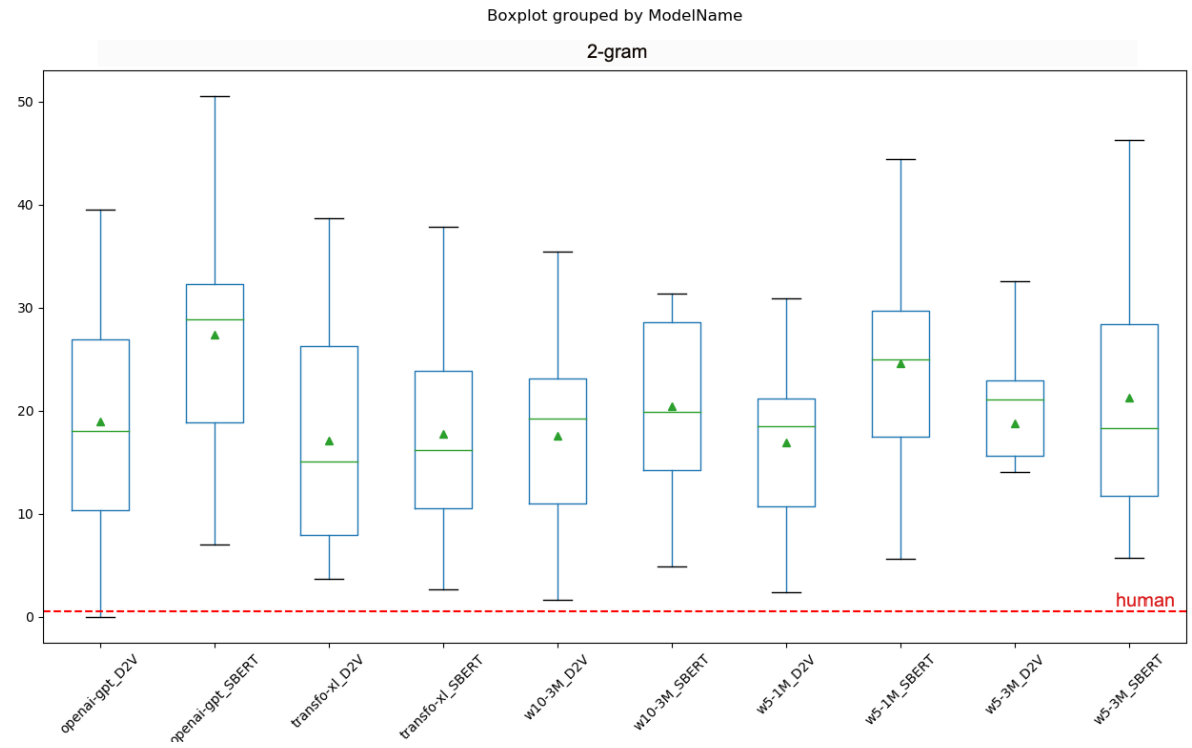



Figure 43: Comparing HA models for Bigram-Prompt overlap percentage for stories. Mean score is shown by the green △.

**Syntactic Style and Complexity – Type Token Ratio.** Studying the Type Token Ratio (TTR) determines the uniqueness in the generated text content. As stated earlier, it is a measure of word usage variety in the generated textual content. The boxplots in Figure 47 shows the properties of the TTR in the textual content of the stories generated by the language models. The TTR values for the generated emails for the different HiGen models are shown in Figure 48. However, the mean scores of the models are not statistically significantly similar to human scores, though in both cases fine-tuned models perform the best.

**Harmonic Mean of Sentence Connectedness.** Inter-sentence cohesion can be calculated using this metric which measures the harmonic mean between the mean and standard deviation of the sentence connectedness ($HM\_sent\_conn$) scores. The measure for every story is calculated by vectorizing the sentence using Sent2Vec embeddings and then measuring the inter-sentence angular deviation. The lower the deviation better is the coherence. Figure 49 shows the results

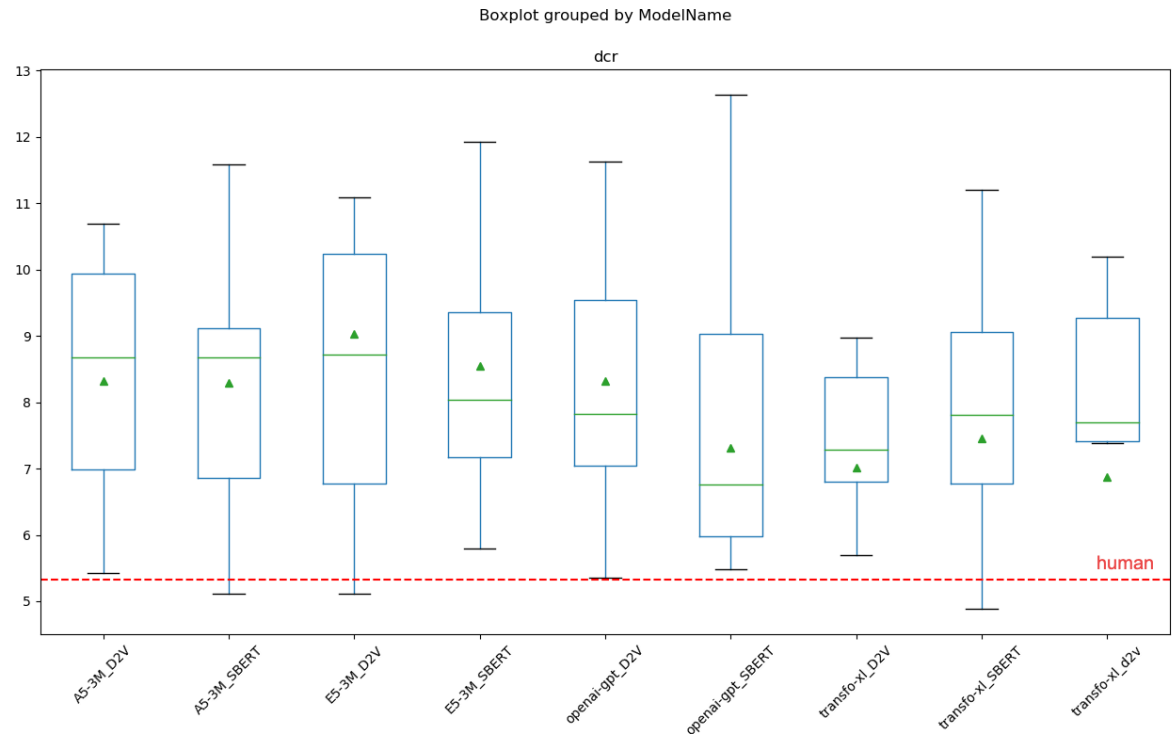



Figure 44: Comparing HA models for Dale-Chall Readability Score for emails. Mean score is shown by the green △.

for $HM\_sent\_conn$ scores through the boxplots of the different models of HiGen architecture on generated stories from prompts. For emails, the scores of the measure are shown in Figure 50.

### 6.4.3 Detection Algorithm for Emails

We frame the problem of differentiating fake emails from legitimate emails as an anomaly detection problem and use a OneClass SVM classifier to differentiate legitimate or human authored emails from synthetic or fake or machine generated emails. Using the OneClassSVM classifier implemented as part of the Python ScikitLearn toolkit, we trained the model on TFIDF count vectors extracted as features from 30,000 Avocado and 30,000 Enron human-written emails. For testing, we used the TFIDF features from 6000 generated Avocado and 6000 Enron emails to determine system performance, along with 6000 legitimate samples. The Table 19 gives a confusion matrix for the generated emails. We consider the human-authored emails as legitimate or Positive (P) class and

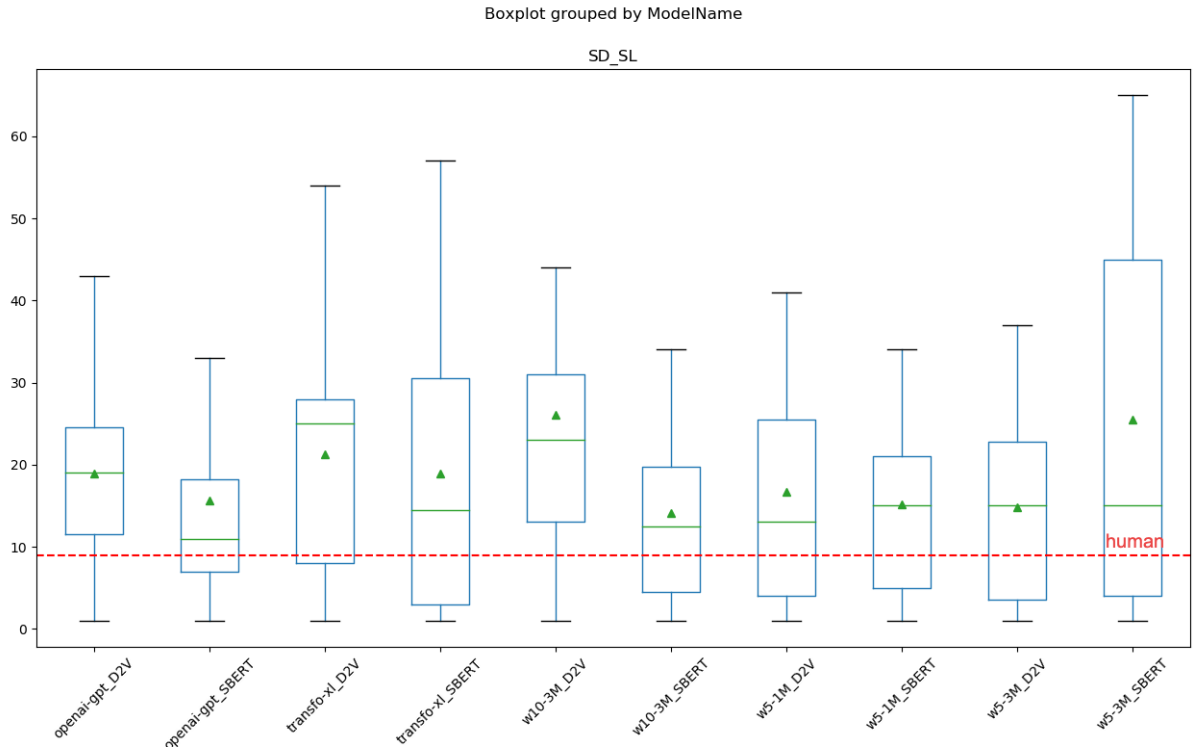



Figure 45: Comparing HA models for $L_{SD}$ for stories. Mean score is shown by the green △.

the synthetic or fake emails as negative (N) class. The precision is 55% and the recall is 40% which is not exceptional performance.

Table 19: Confusion matrix for Synthetic Email Detection

| | **Actual** | |
|---|---|---|
| **Predict** | P | N |
| P | 4281 | 3453 |
| N | 6342 | 3924 |

## 6.5 Qualitative Analysis

This section provides a qualitative evaluation of the textual instances provided by the different language models as part of the proposed HiGen architecture.

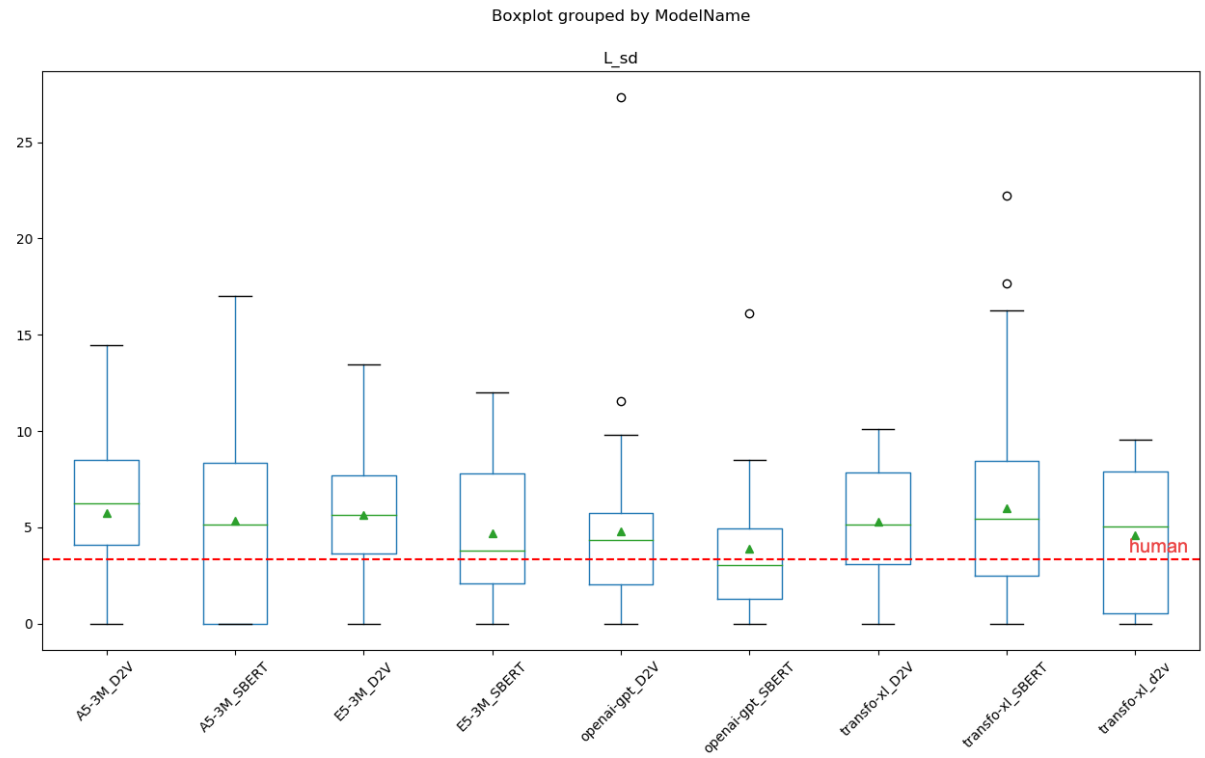



Figure 46: Comparing HA models for $L_{SD}$ for emails. Mean score is shown by the green $\triangle$.

### 6.5.1 Generated Samples of Stories

We include some selected samples generated by different trained HiGen models. Examples 21 and 22 show stories generated from prompts by the OpenAI-GPT pre-trained model with SBERT embeddings. The samples in Tables 23 and 24 have been generated using the Transfomer-XL language model with D2V sentence embeddings. Finally, Tables 25 and 26 show examples with fine-tuned language models and sentence-level embeddings S-BERT and D2V. In Table 20, we compare the stories generated by different models for a given similar starting story prompt.

### 6.5.2 Generated Samples of Emails

We include the email samples generated by different variations of the HiGen language models. The samples generated by the pre-trained models OpenAI-GPT and Transformer-XL with the two sets of sentence embeddings, SBERT and D2V are presented in the Boxes 6.1 and 6.2 respectively.

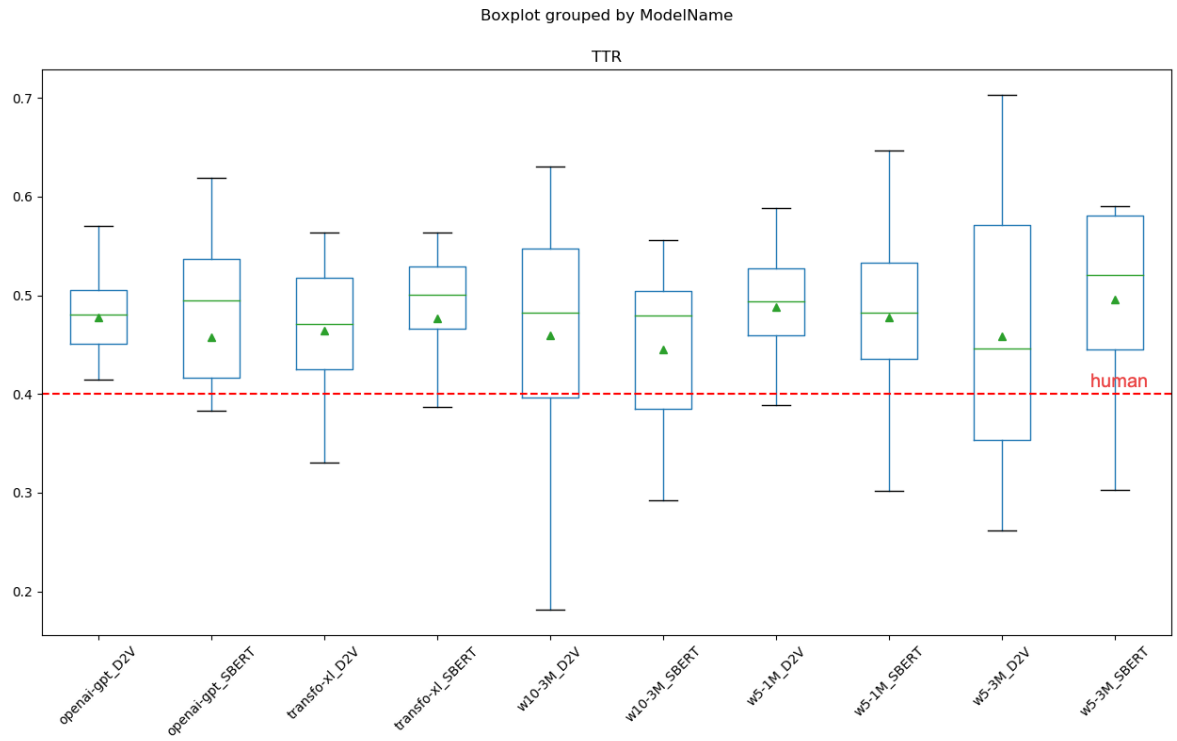



Figure 47: Comparing HA models for Type Token Ratio for stories. Mean score is shown by the green △.

Table 20: An example showing a prompt and the stories generated using the generative-predictive model combinations for the HiGen architecture.

| | |
|---|---|
| **Prompt** | Whilst restocking a vending machine, a janitor finds something horrific jammed inside. |
| **TX-D2V** | On one occasion, he entered a bar and was beaten by a woman,<br>who struck him in the chest with a beer bottle, and he was taken to hospital<br>where he was diagnosed with acute respiratory distress syndrome. On 15 October 2006,<br>the US Court of Appeals for the Second Circuit upheld the conviction of Christopher<br>for “attempting to cause serious injury in a bar in Oakland, California ”.’ [...] |
| **OG-SBERT** | Whilst restocking a vending machine, a janitor finds something<br>horrific jammed inside.’, ”’this isn’t right! ’this isn’t right!”<br>The way he was trying to read my face and his eyes made me realize he must know<br>who I was. “He shoved the button on the remote, and a familiar voice answered. ”.<br>‘No - she didn’t come over to get me. ” [...] |
| **w5-3M-SBERT** | ”wow , wow !” “hey , are you ...” “just stop ,<br>I’m not going to ruin this too much!” “kk- I’m really getting hurt ...”<br>“oh right ...” “what ?” “okay what ’s your name?”, “ I really hope you are okay ,<br>I’ve been working on this for a long time.” [...] |

Similarly, we also include the samples generated by the fine-tuned generative models, E5-3M and A5-3M with the different types of sentence embeddings in the boxes 6.3 and 6.4.

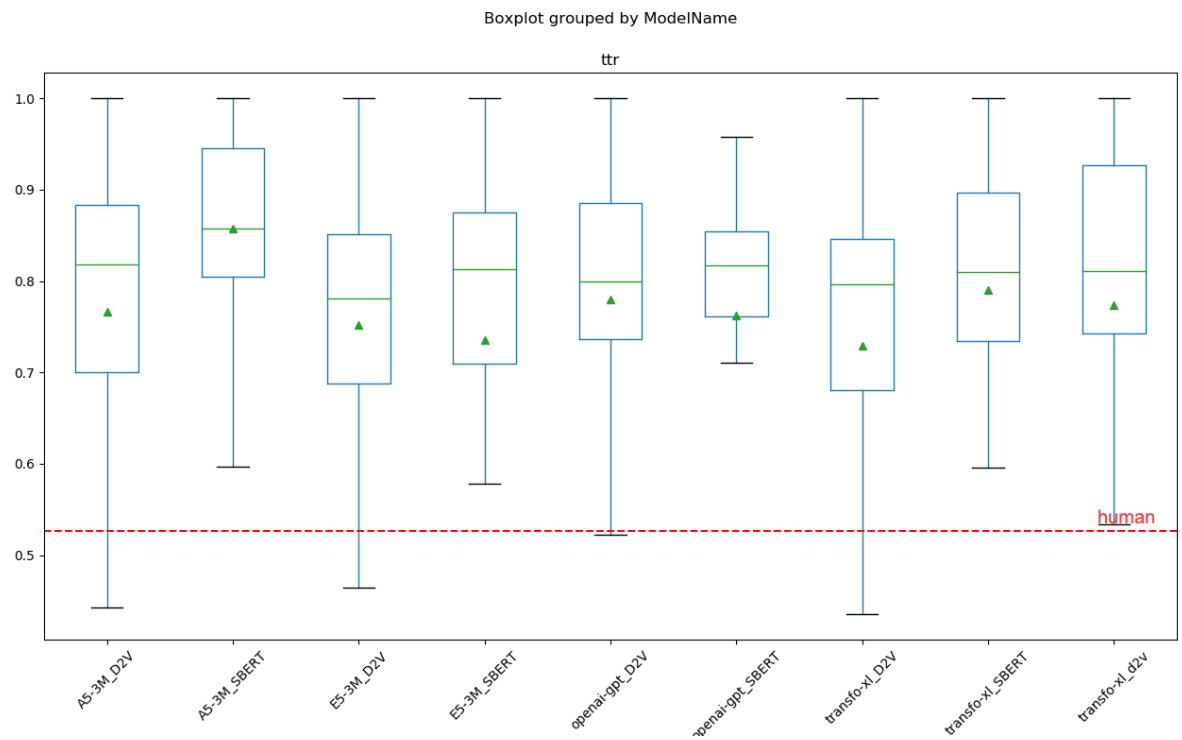



Figure 48: Comparing HA models for Type Token Ratio for emails. Mean score is shown by the green △.

Table 21: An example showing a prompt and a generated story using the **OG_SBERT** model

| | |
|---|---|
| **Prompt** | You have the ability to reverse time by 5 seconds,<br>this allows you to always say and do the right things, until the day<br>you meet the one person immune to the time distortion. |
| **OG_SBERT** | “ The first recorded instance of a person having a vision of the future<br>was in the case of the young boy, who had a vision<br>of the future. The child was born in the morning of November 4, 1889.<br>The child was raised by his mother, and was raised by her brother.<br>The first recorded case of a person having<br>vision of the future was in the case of a young man who had a vision of the future. [...] |

Table 22: An example showing a prompt and a generated story using the **OG_SBERT** model

| | |
|---|---|
| **Prompt** | Write a story that seems like a cliffhanger untill you reread the first line. |
| **OG_SBERT** | I like the characters in this book a lot. it’s a lot like how i’ve always<br>liked my dad, except it’s a lot like how i’ve always enjoyed my life.<br>’now... i know it’s been a long time,’he said. ’too long,’said mort.<br>’let’s just say i didn’t know you were there, too. mort did not want to appear hostile,<br>but he was far too far away to try any sort of casual introduction<br>with this level of scrutiny, so he held up one finger. [...] |

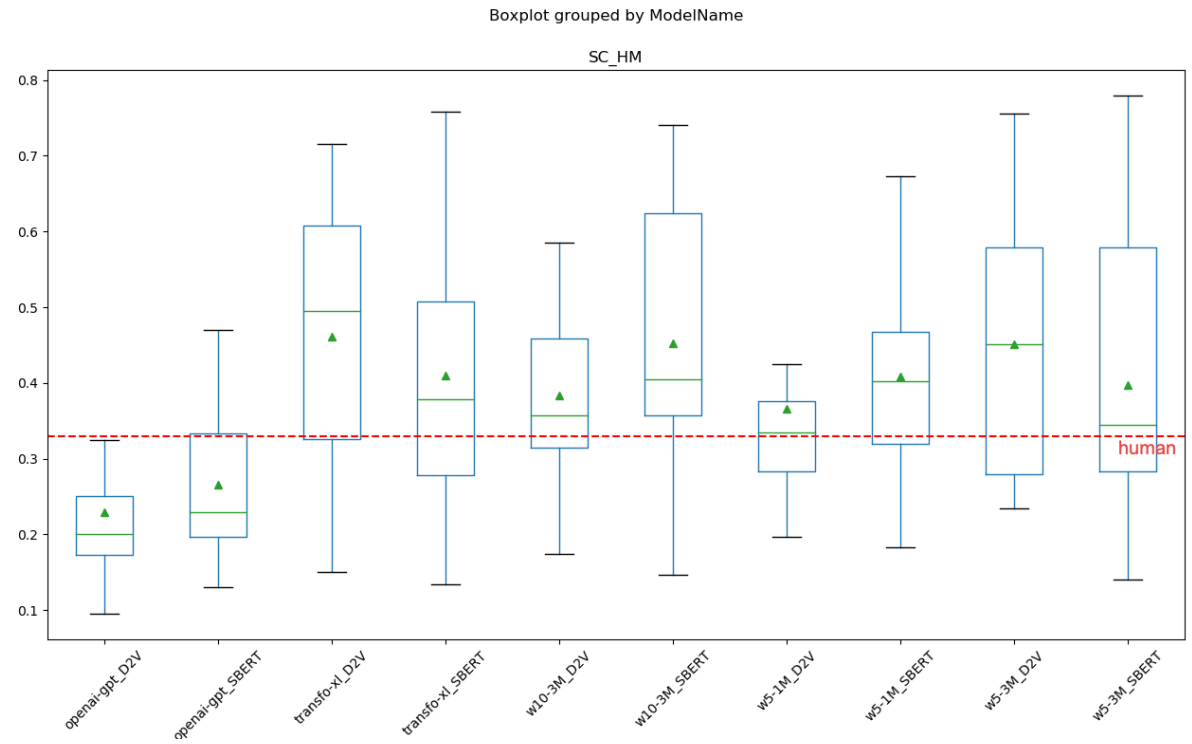



Figure 49: Comparing HA models for Harmonic Mean of Sentence Connectedness for stories. Mean score is shown by the green △.

Table 23: An example showing a prompt and a generated story using the **TX_D2V** model

| | |
|---|---|
| **Prompt** | You live in a world where people's shadows show who they truly are at their core.<br>Some shadows look like monsters, some look like animals.<br>You are the only person in the world with no shadow. |
| **TX_D2V** | " In 2003, Time magazine included him in their list of the 100 most influential people<br>in the world. On one occasion, a man, sitting in his car on the street,<br>was struck by a car driven by a horse, and died in the accident.<br>He's got an innate sense of humor, a sense of heroism that allows him<br>to speak something very real in all situations. Though ostensibly a parody<br>of the archetype, he also looks exactly like Johnny, the " stupid gangster " from Scarface. [...] |

Table 24: An example showing a prompt and a generated story using the **TX_D2V** model

| | |
|---|---|
| **Prompt** | Turn a well known songs lyrics into a short story |
| **transformer-xl_D2V** | On one occasion, he entered a radio room and started singing<br>"The Prayer ", a hymn from the Catholic soundtrack to a film he was making called The Passion of the Christ.<br>After the death of his first wife In 1958, after a year of marriage, he married Anne Evans,<br>a teacher at the University of Pittsburgh. The song was released on December 2, 1983 by Sire Records,<br>as the lead single from the album. "Discography Studio albums You 'll Never Know (1973)<br>" Get Loud (Live in Las Vegas)" (1974) You 'll Never Know (1975) "The Prayer a la Mode" (1976) I Am Not Alone<br>" I Am Not Alone " is a song recorded by American singer Madonna<br>for her fourth studio album American Life (1983) [...] |

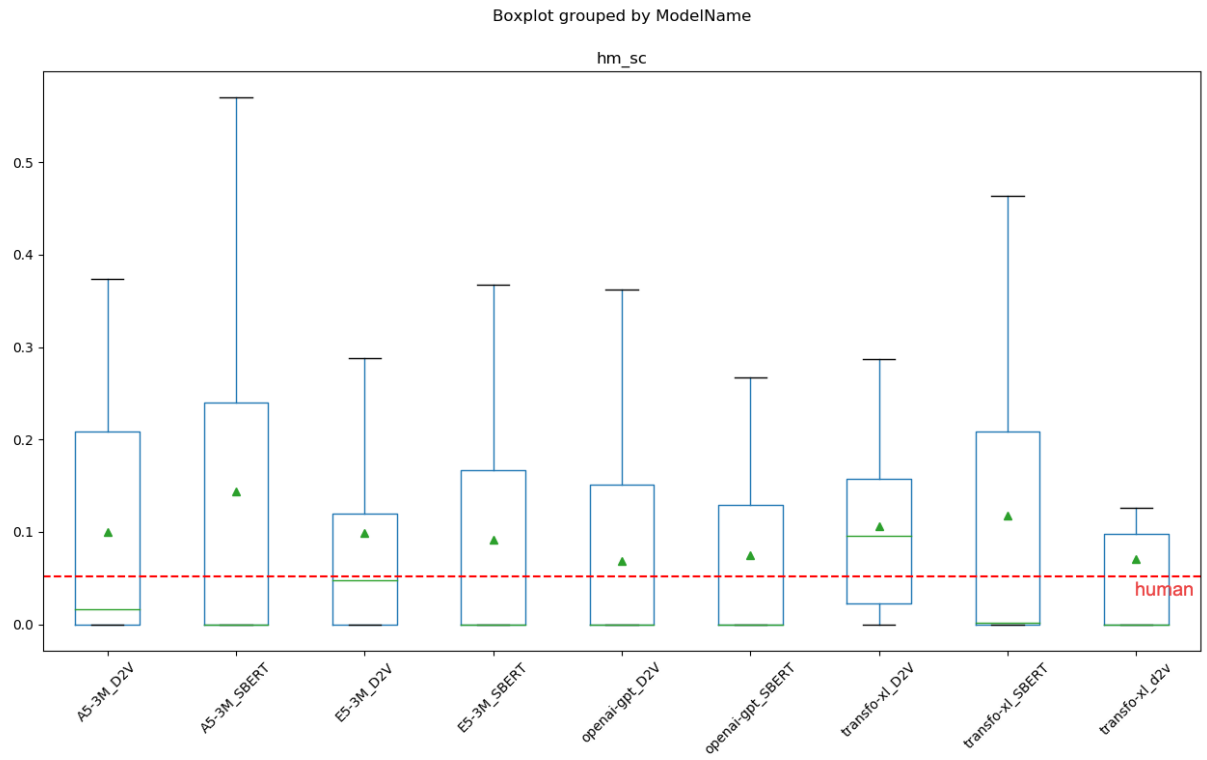



Figure 50: Comparing HA models for Harmonic Mean of Sentence Connectedness for emails. Mean score is shown by the green △.

Table 25: An example showing a prompt and a generated story using the **w10-3M** model with **SBERT** embeddings

| **Prompt** | Pitch a gritty reboot for a children's show . |
|---|---|
| **w10-3M_SBERT** | "I will never reveal what you are today until now.<br>sappy responded , smugly condescendingly . she asked .<br>what do you want ?" "and i think you do too."<br>" i think you have it in your head."<br>" i know." "i think so." "i think so." "i do n't think so." [...] |

Table 26: An example showing a prompt and a generated story using the **w10-3M** model with **D2V** embeddings

| **Prompt** | In 200 words or less, write a well-known villain as a hero,<br>but do not tell us who they are. |
|---|---|
| **w10-3M_D2V** | They came for me, they came for my family. *i 'm sorry*, i said out loud,<br>something was said, in my darkest hour, i was running for my life.<br>I did n't had time to listen, but i knew what i was running from. [...] |

Table 27: An example showing a prompt and a generated story using the **w5-1M** model with **SBERT** embeddings

| | |
|---|---|
| **Prompt** | It used to keep me up at night, now it helps me sleep. |
| **w5-1M_SBERT** | Every time I wake up, the old lady in front of me looks up at me.<br>She's staring at me. She steps into my room like she is waiting for me.<br>She 's staring at me. I can't open my eyes.<br>I ca n't swallow it. I'm going to burn it all. [...] |

Table 28: An example showing a prompt and a generated story using the **w5-1M** model with **D2V** embeddings

| | |
|---|---|
| **Prompt** | in a world where the all manufacturing processes have been completely automated,<br>there is an apocalypse. Gradually, the machines become self aware,<br>and slowly realise that the humans are gone. |
| **w5-1M_D2V** | so by their last minute- many people think it would be harder to make the machines,<br>then they would have solved the problems of who makes the machines , right ?<br>these machines are just such of them . their machines are getting faster,<br>and they are getting faster. so they are getting more advanced,<br>and more powerful, and they are getting more efficient.<br>he's unstoppable and the slave human race has become, but you have to remember<br>this the right way to describe his accomplishments : he 's a hard man, or at least,<br>looks so hard , to the human people. it 's just like my job in a cube . [...] |

**Box 6.1: Samples generated by pre-trained OpenAI-GPT by varying sentence embeddings**

**A1. Conditional Sampling with SBERT embeddings:**
*Email Subject: new link for all-employee meeting*
Attend ... details go for later
**A2. Conditional Sampling with D2V embeddings:**
*Email Subject: subject:aether on thick vs thin*
book ? What kind u say ... ? do yaki wan na read these

**Box 6.2: Samples generated by pre-trained Transformer-XL by varying sentence embeddings**

**B1. Conditional Sampling with SBERT embeddings:**
*Email Subject: gmm 26 oct 01*
No Exploration Informationreceived 41Jul2001, or receipt any natural turnbacks sent through activity recovery
**B2. Conditional Sampling with D2V embeddings:**
*Email Subject: counterparty approval list for 8/30/00*
2 or new is therefore just needing documents.Step hoch Triam.
Need 5 additional examples prepared

**Box 6.3: Samples generated by fine-tuned Enron_512_355M by varying sentence embeddings**

**C1. Conditional Sampling with SBERT embeddings:**
*Email Subject: re: financial settlements*
Not sufficient balance if markets understand seasonal preferences
**C2. Conditional Sampling with D2V embeddings:**
*Email Subject: latest & greatest q & a document*
which various have assisted . ” ?

**Box 6.4: Samples generated by fine-tuned Avocado_512_355M by varying sentence embeddings**

**D1. Conditional Sampling with SBERT embeddings:**
*Email Subject: Re: AvocadoIT*
Yes, the meeting will start at 3.00pm.
**D2. Conditional Sampling with D2V embeddings:**
*Email Subject: subject:re: are you here today?*
thanks tired hOL MAR hour : (

# 7 Evaluation of Human Performance

There exists little agreement in the way a generative system should be evaluated [49]. However, most researchers consider human study to be the best possible practice to evaluate the task-based effectiveness of generative systems [94, 49]. In this section, we discuss in detail the human evaluation study we performed on the generated textual content – stories and emails. We explain the study protocol design and the data cleaning, followed by a description of our analysis on the results observed from the study responses. In Section 7.3.3, we provide a brief overview of the human participants' judgement based on topic analysis of the stories and the average confidence level for emails. We also study the correlation between the human personality traits and web usage skills with the detection and confidence levels of the participants in Section 7.4. Finally, we analyze the correlations between the automated metrics and human scores in Section 7.5.

## 7.1 Study Protocol and Data Cleaning

We conducted our Survey after receiving approval from the university's Institutional Review Board for the study. We invited all students, faculty and staff members and professionals to participate in the study. We also posted the link on Amazon Mechanical Turk and invited Turkers to participate in our studies. We explained in detail the tasks to the participants and mentioned that the task takes around 30-40 minutes to complete.

We used the data collection platform, Alchemer,[48] to host our survey questionnaire and collect responses anonymously from the participants. The site also allows us to track the amount of time taken by the participants to complete each question. The survey starts with 44 personality trait based questions,[49] followed by the some questions to understand the participants web usage skills and familiarity with common web-based terms [44]. Prior to the task of evaluating stories, the participants were asked some questions on their fluency in English and whether they have any experience in writing creative English content. Each participant was asked to evaluate the nature

[48] `https://www.alchemer.com/`

[49] http://www.uoregon.edu/ sanjay/bigfive.html#where

and textual quality of four email instances and four story instances. For the email evaluation task, two emails were human written emails selected randomly from 12 such instances and two machine generated emails were sampled randomly from the emails generated by the HiGen architecture. For the samples included in the evaluation of the stories, two of the four stories were human written (true label is "Real") – selected randomly from five graded essay samples[50] and two were machine generated stories (true label is "Fake") – selected randomly from five synthetic samples generated by the HiGen architecture. During the story evaluation task, each participant was given a prompt and a story pair. For evaluating the emails, we include description of the situation in which the communication between the sender and receiver takes place to provide more context to the study participants.

We received a total of 207 responses over the course of five days ($29^{th}$ October, 2020 to $24^{th}$ November, 2020), of which 97 responses were complete. Of these, 63 were from the Amazon Mechanical Turk (**AMT**) and 34 were from the regular participants recruited using the university-wide recruiting (**Reg**). Based on the participants' response to the two attention-based control questions in the email and story tasks, we filter out 70 (40 **AMT** + 30 **Reg**) responses for stories and 62 (39 **AMT** + 23 **Reg**) responses for emails. We also impose additional checks on the responses of the Amazon Mechanical Turkers like matching a Survey ID, checking the quality of responses (i.e., if the same response has been repeated across the survey), one word answers to the descriptive questions. The average time taken by AMT was 20.16 minutes, a statistically significantly lower time than the regular participants (35.91 minutes).[51] In the following sections, we analyze the human personality traits and web usage skills, followed by their performance on evaluation tasks and summarize the key findings for each genre of textual content.

[50] https://patternbasedwriting.com/elementary_writing_success/elementary-writing-samples-middle-school-writing-examples-sample-essays/

[51] Using an unpaired two-samples t-test, we get $p$-value = 0.003426.

## 7.2 Human Personality Traits and Web Usage Skills

We study the raw scores of the Turkers (AMT) and the regular group of participants (Reg) for the different human personality traits and web usage skills in the following sections.

### 7.2.1 Human Personality Traits

The Big 5 personality traits are Extroversion (E), Agreeableness (A), Conscientiousness (C), Neuroticism (N) and Openness to Experience (O). We refer the readers to [7] for more details on how these are calculated and the definitions.[52] Using box plots in Figure 51, we show the variation in the raw scores of the big 5 personality traits for the three groups of participants – AMT, Regular and overall. Table 29 shows the statistical significance ($p-$values) between the scores of the personality traits for the two groups (AMT and Reg) calculated using the unpaired independent two-samples t-test. The table also shows the mean personality trait scores. We see that for AMT and Regular participants, there exists no statistically significant difference in their personality traits based on their $p-$values.

Table 29: Mean scores of the participant groups on Human personality traits. The $p$-values are shown in parentheses.

| | **E** | **A** | **C** | **N** | **O** |
|---|---|---|---|---|---|
| **AMT** | 24.6 | 29.3 | 29 | 25.4 | 33.4 |
| **Reg** | 24.6 (0.974) | 30.3 (0.235) | 30.05 (0.197) | 25.4 (0.984) | 33.2 (0.890) |

### 7.2.2 Web Usage Skills

We ask our study participants to rate their knowledge of some common web-based terms like "Advanced Search" (Adv_Search), "Preference Settings" (Pref_Sett), "PDF", "SpyWare" and "Wiki". These terms have been defined in [44]. The participants were asked to rate their web usage skills on a Likert scale of [1, 5], with 1 being 'No understanding' to 5 referring to 'Full understanding' of the term. We present the boxplots of the web usage skill scores in Figure 52. Similar to human personality traits, the mean scores on the familiarity with the different web terms has been shown in

[52] https://positivepsychology.com/big-five-personality-theory/

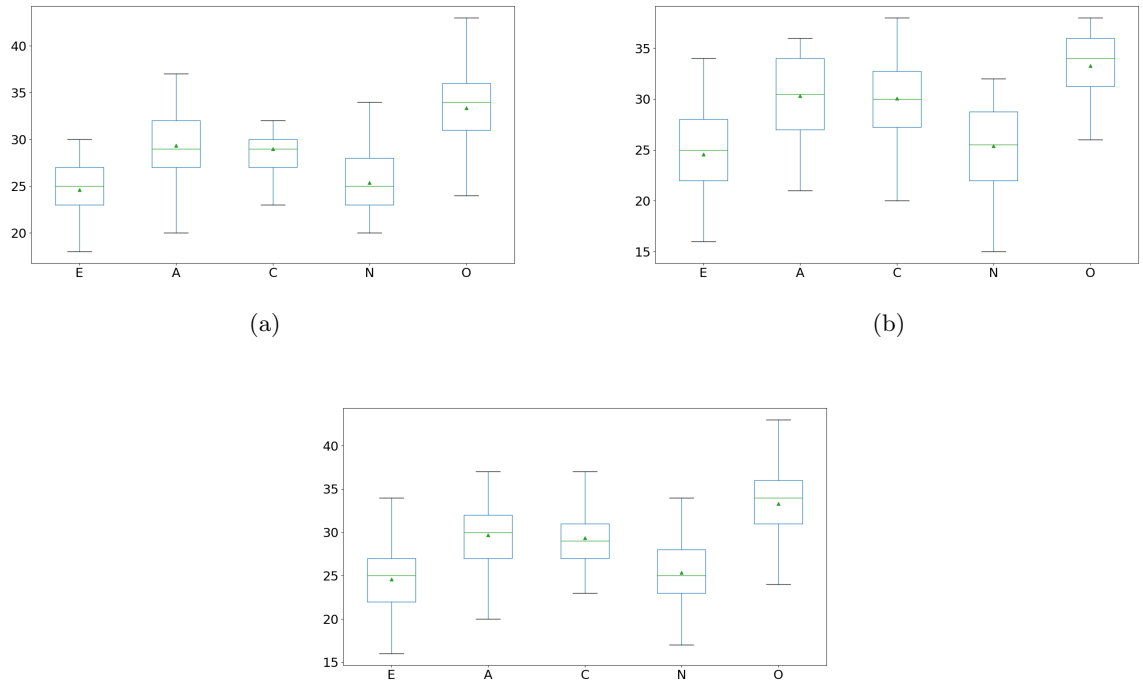



Figure 51: Boxplots showing scores of participants on human personality traits for (a) AMT (b) Regular (c) Overall. Mean score is shown by the green △.

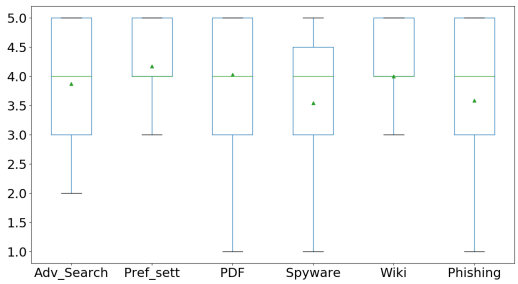
Table 30. It also shows the $p-$values between the different scores for AMT and Regular participant groups. We observe that, except for the familiarity with the term 'PDF', there exists no significant statistical difference between the mean familiarity ratings with the other web terms for the groups AMT and Regular.

Table 30: Mean scores of the participant groups on Web Usage skills. The $p$-values are shown in parentheses.

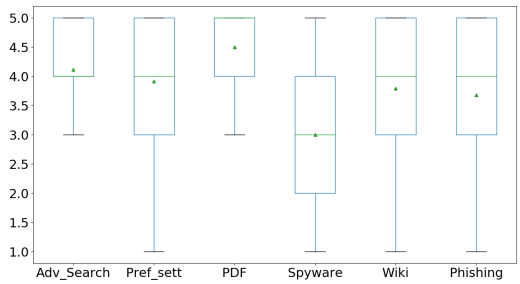
| | **Adv_Search** | **Pref_Sett** | **PDF** | **Spyware** | **Wiki** | **Phishing** |
|---|---|---|---|---|---|---|
| **AMT** | 3.88 | 4.17 | 4.03 | 3.54 | 4 | 3.58 |
| **Reg** | 4.12 (0.276) | 3.91 (0.013) | 4.5* (0.009) | 3 (0.067) | 3.79 (0.426) | 3.67 (0.741) |

## 7.3 Task-based Performance Analysis

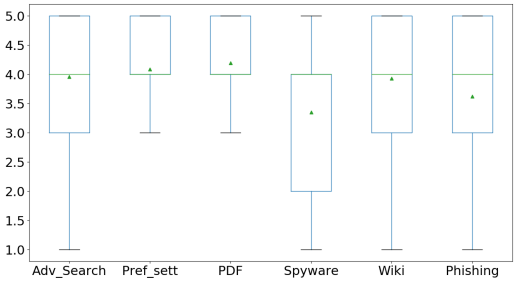
The participants were given two major tasks to perform:




(a)




(b)




(c)

Figure 52: Boxplots showing scores of participants on Web Usage skills for (a) AMT (b) Regular (c) Overall. Mean score is shown by the green △.

- Task A: Given a set of emails, with a scenario in which the sender sent the email to the receiver. These emails resemble organizational emails, some of which are real and some are fake. The participants were asked to label the emails as fake or real and also rate how confident they are about their decision.

- Task B: Similar to the emails, the participant was presented with stories generated by automated machine learning algorithms from a prompt. Along with the fake or machine generated emails, they were also presented with stories written by humans (real) from a given prompt or topic. The participants were asked to identify the given story as real or fake and also to specify if the story stays on topic or not.

For the both the tasks, each participant was given a total of eight questions (4 for each task, and each set consisting of 2 real instances and 2 fake instances). Apart from identifying the instance as

real or fake, each participant was asked to rate the text on fluency and its syntax or organization on a Likert scale of 1 to 5, with 1 being the lowest score and 5 being the highest. We asked the participants to rate fluency as per the following instructions: a low fluency means that the author is repeating oneself multiple times, has limited ability to link simple sentences, and is unable to convey a basic message. Moderate fluency scores mean that the author writes text more or less coherently and is effectively able to convey a message. High fluency means there is no hesitation in content and the text written is appropriate and develops fully and appropriately on the topic. For evaluating the syntax/organization of the text, we asked the authors to rate the text based on the arrangement of words and phrases and whether they create well-formed, grammatically correct sentences. The participants were asked not to consider spelling errors, but to evaluate the text based on its grammatical correctness.

The effectiveness of the proposed content generation system – HiGen is evaluated using the following metrics – **detection rate** (as %) of the participants, along with average **scores on fluency and syntactical organization** (i.e., syntax) of the generated content. We also include the observations on the participants' level of confidence. **Detection rate** (DR) is the proportion of instances that are detected correctly, real detected as real and fake detected as fake. DR is defined in the Equation 9.[53] **Confidence level** (CL) is the degree of confidence that participants have in their answers [7].

$$DR = \frac{TP}{TP + FN} \tag{9}$$

### 7.3.1 Evaluating Stories

The average detection rate, fluency and syntax scores on the generated and human-written stories are reported in Table 31. We include the results of the participants from each group – AMT and Reg separately, as well as together. Figure 53 shows the results for human evaluation using overall detection rate along with real and fake detection rates i.e., the proportion of correctly detected real and fake samples respectively. Along with DR, we also show the same for fluency and

[53]TP = True Positive or instances that are correctly classified as positive or real and FN = False Negative or instances that are incorrectly classified as negative or fake

syntax/organization scores of the textual content in Figures 54 and 55. We compare the detection rates between the human participants from AMT and Regular groups based on their statistical significance, we see that the DR of AMT participants on fake instances are significantly lower than the regular participants in case of stories as seen from the $p$-values in Table 31.

Table 31: The Detection Rate, Fluency and Syntax scores on Real VS. Fake Stories. **DR**: Detection Rate, **RDR**: Real Detection Rate, **FDR**: Fake Detection Rate, **Flu**: Fluency, **Syn**: Syntax. The $p$-values are shown in parentheses.

| | **DR (%)** | | | **Average Scores** | |
|---|---|---|---|---|---|
| | **RDR** | **FDR** | **DR** | **Fluency** | **Syntax** |
| **AMT** | 68.04 | 31.76 | 49.90 | 3.34 | 3.32 |
| **Reg** | 65.56 (0.797) | 63.03* (0.011) | 64.28* (0.057) | 2.97 (0.145) | 2.84 (0.098) |

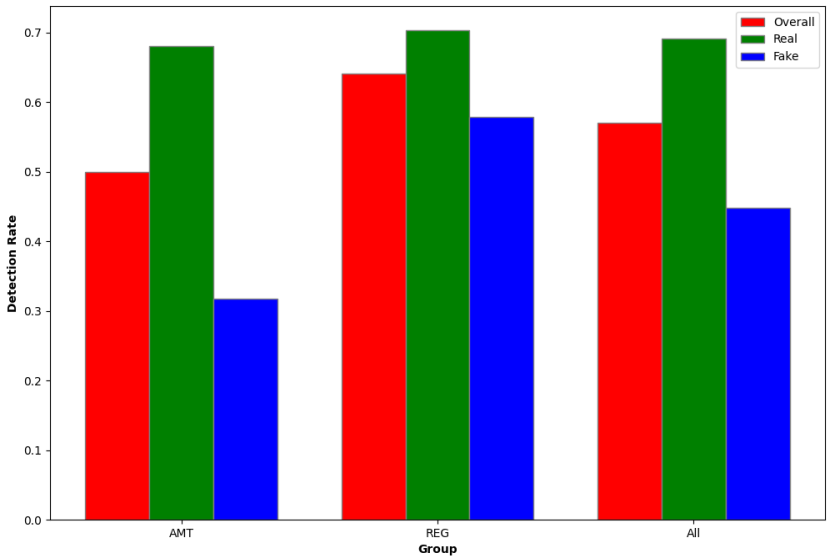



Figure 53: Human Detection Rate for stories

### 7.3.2 Evaluating Emails

Like stories, we analyze the average detection rate, fluency and syntax scores on the generated and human-written emails. The results of the human evaluation are reported in Table 32. Here also, we report the observations for the two groups, AMT and Reg, along with the overall results.

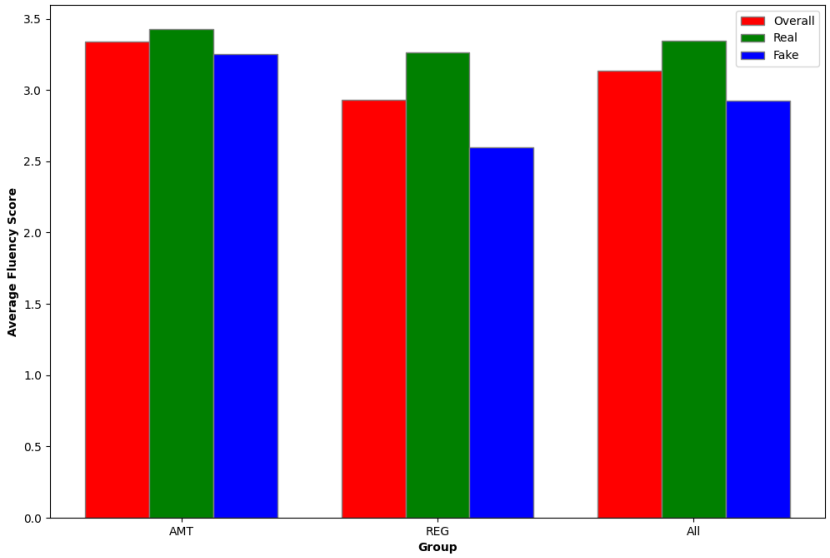



Figure 54: Average Fluency scores for stories

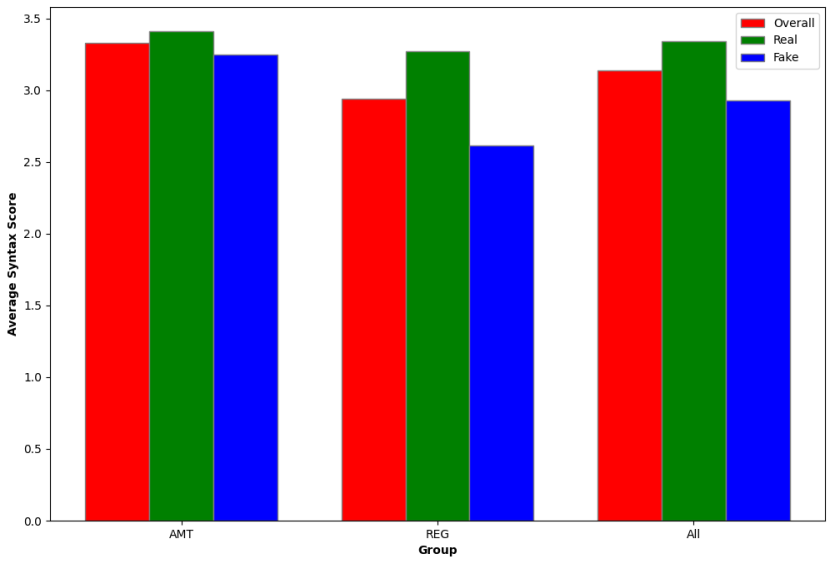



Figure 55: Average Syntax scores for stories

Figure 56 shows the results for human evaluation using overall detection rate along with real and fake detection rates i.e., the proportion of correctly detected real and fake samples respectively.

Along with DR, we also show the same for fluency and syntax/organization scores of the textual content in Figures 57 and 58. The $p-$values shown in Table 32 are used to evaluate the statistical significance of the detection rates of the two groups of participants. Using the unpaired two-samples t-test, we see that there exists no statistically significant difference between the groups' DRs (also, RDR and FDR). Similar is the case for fluency and syntax scores.

Table 32: The Detection Rate, Fluency and Syntax scores on Real VS. Fake Emails. **DR**: Detection Rate, **RDR**: Real Detection Rate, **FDR**: Fake Detection Rate, **Flu**: Fluency, **Syn**: Syntax. The $p$-values are shown in parentheses.

| | **DR (%)** | | | **Average Scores** | |
|---|---|---|---|---|---|
| | **RDR** | **FDR** | **DR** | **Fluency** | **Syntax** |
| **AMT** | 66.38 | 21.40 | 43.89 | 3.78 | 3.71 |
| **Reg** | 50.90 (0.194) | 22.01 (0.953) | 36.45 (0.427) | 3.58 (0.304) | 3.41 (0.197) |

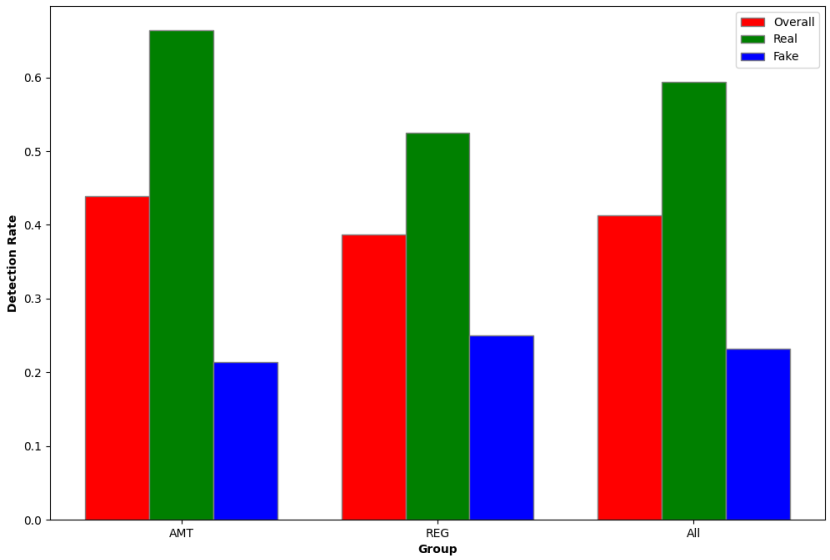



Figure 56: Human Detection Rate for emails

### 7.3.3 Human Judgement Analysis

We asked the participants' to rate their confidence in their judgement on a Likert scale of [1,5] with 1 being the lowest and 5 being the highest score. Based on the rating, we report the overall average

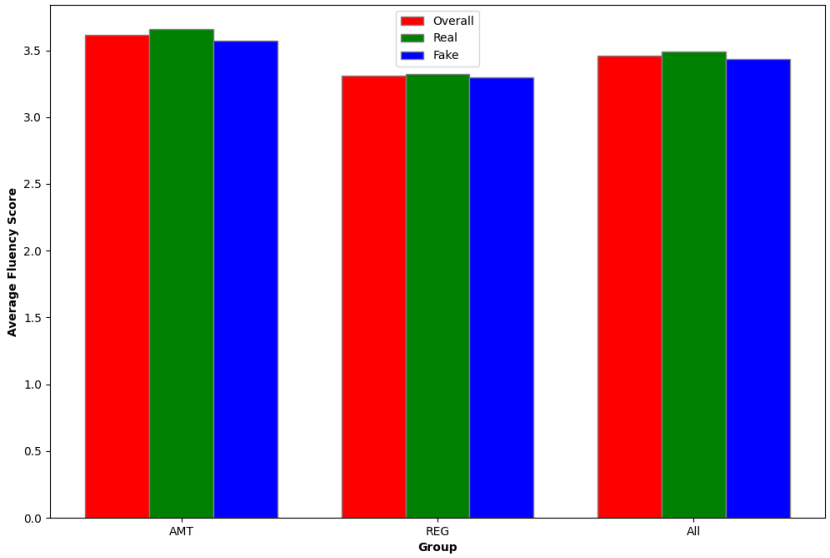



Figure 57: Average Fluency scores for emails

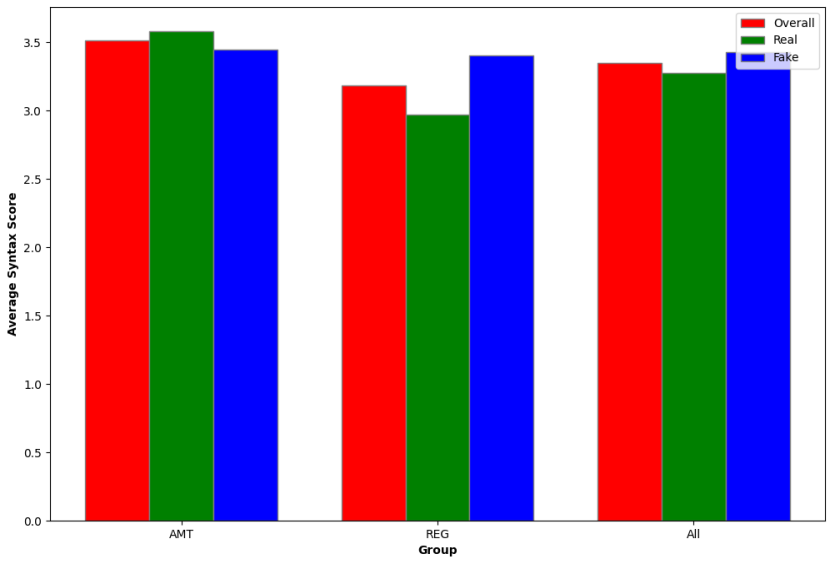



Figure 58: Average Syntax scores for emails

confidence level of the participants irrespective of the number of correct or incorrect predictions. Participants may express a different level of confidence when they correctly classify an instance

compared to the cases when they are wrong. To analyze the participants' confidence levels, we calculated two sets of average confidence levels for each participant – i. for correctly classified instances (true negative and true positive), and ii. for incorrectly classified instances (false negative and false positive). Our results for the participants' confidence analysis are shown in Table 33. Using the unpaired two-samples t-test for statistical significance, we see that the average confidence levels are not statistically significantly different between the two participant groups – AMT and Regular.

Table 33: The scores for average confidence level analysis based on human judgement. The $p$-values are shown in parentheses.

| | **Nature** | **Average Confidence Level** | |
|---|---|---|---|
| | | Marked Real | Marked Fake |
| **AMT** | Real | 4.1 | 4.26 |
| | Fake | 4.14 | 3.73 |
| **Reg** | Real | 4.32 (0.254) | 3.8 (0.241) |
| | Fake | 3.89 (0.308) | 3.78 (0.933) |

We also asked the participants to rate whether a story, irrespective of its nature (real/fake), is on topic. Table 34 shows the proportion of the real and fake stories that is related to the topic, in this case the writing prompt as opposed to the proportion that is not. We present the results separately for each group – AMT and Reg.

Table 34: The scores for topic-story relatedness analysis based on human judgement. The $p$-values are shown in parentheses.

| | **Nature** | **Stays on topic?** | |
|---|---|---|---|
| | | Yes (%) | No (%) |
| **AMT** | Real | 70.18 | 29.81 |
| | Fake | 57.38 | 42.61 |
| **Reg** | Real | 82.70 | 17.29 |
| | Fake | 38.47 | 61.52 |

### 7.3.4 Reasoning Analysis

Besides asking participants to tag each email as legitimate or fraudulent, we also asked them to write down their reasoning for their choice. It helps us to have a better understanding of the clues that guided the participants to their final decision. Therefore, we extract the reasoning separately

for each group of participants and their chosen tag. Figure 59 provides the WordCloud-based distributions of the reasoning provided by the participants for their predicted nature of the textual content.


(a) (b)

(c) (d)

Figure 59: Comparing participants' reasoning for choice using Word Clouds where (a) AMT for Real (b) AMT for Fake (c) Reg for Real (d) Reg for Fake.

## 7.4 Studying human personality and web usage skills

In this section, we study the correlation of the Big 5 personality traits of the human participants with their detection rate and confidence levels. We present our observations on the overall participant

performance as well as separately on Amazon Mechanical Turkers (AMT) and Regular (Reg) groups.

### 7.4.1 Correlation with human personality traits

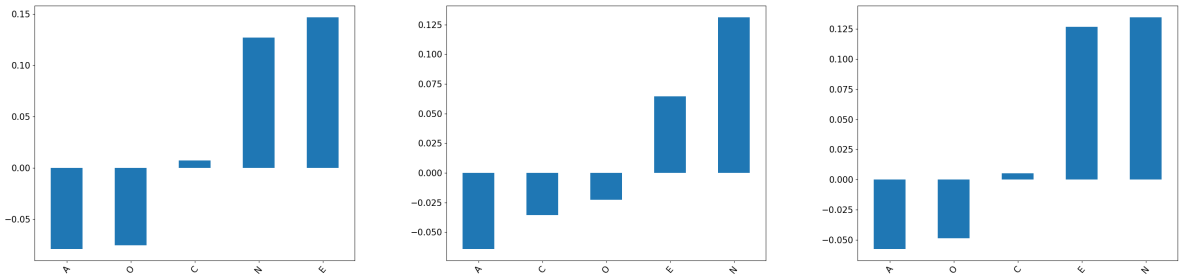
For the purpose of this analysis, we look at the regression analysis between the detection rate or confidence level of the participants with the human personality (hp) traits.

For the analysis of the **detection rate**, we measure the proportion of correctly identified instances by the participants and correlate them with their normalized personality trait scores. We use the Ridge Regression analysis algorithm from the Scikit-Learn library. The results of the analysis are shown below, with the regression coefficients shown in Figure 60 for the overall, AMT and Regular groups respectively. Equations in10 show the regression equations for the same. Here, DR refers to detection rate. We also calculate the $R^2$ and adjusted $R^2$ scores for the different analysis scenarios for detection rate – for AMT participants (0.076, -0.034); regular participants (0.044, -0.140) and overall (0.052, -0.012).

The analysis with the **confidence level** scores is the same. The correlation coefficients shown in Figure 63 along with the values shown in Equation 13 describe the relationship between the confidence levels of the different groups of participants (AMT, Regular and overall) with the human personality traits. We also calculate the $R^2$ and adjusted $R^2$ scores for the different analysis scenarios for confidence level – for AMT participants (0.203, 0.004); regular participants (0.115, -0.287) and overall (0.043, -0.086).

$$\begin{aligned} DR_{All} &= 0.135 * N + 0.127 * E + (-0.057) * A + (-0.048) * O + 0.005 * C \\ DR_{AMT} &= 0.147 * E + 0.127 * N + (-0.079) * A + (-0.075) * O + 0.007 * C \\ DR_{Reg} &= 0.131 * N + 0.065 * E + (-0.064) * A + (-0.036) * C + (-0.023) * O \end{aligned} \tag{10}$$

We also compare the correlation of the detection rate and confidence levels of the participants separately for each genre of the task with their human personality traits. Figure 61 shows the regression correlation coefficients of the human personality traits with the detection rate (DR) on stories for the AMT and Regular groups of participants followed by the overall rates. The regression model



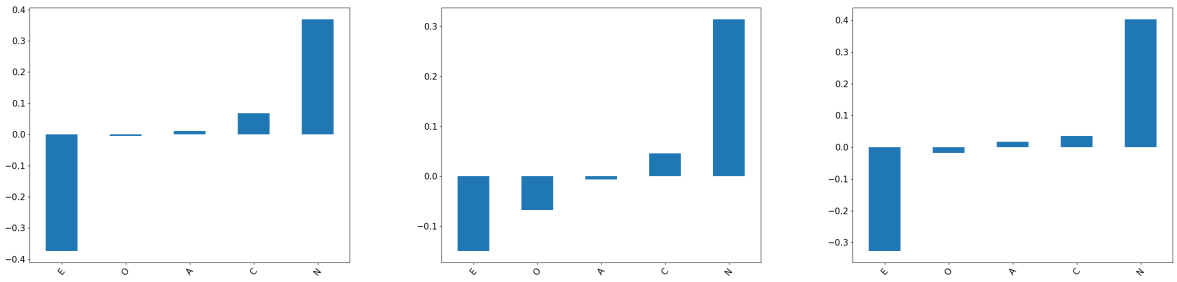

(a) (b) (c)

Figure 60: Comparing Regression Analysis Coefficients for human personality traits with Detection Rate on emails and stories for (a) AMT (b) Regular (c) Overall.

equations for the three groups for detection rate on stories are shown in Equation 11. The same has been done in case of emails, the correlation coefficients calculated using the Ridge Regression analysis are shown in Figure 62 and their relation with the DR is shown in Equation 12. The $R^2$ and adjusted $R^2$ scores for the detection rates on stories – for AMT participants (0.295, 0.188); regular participants (0.184, 0.014) and overall (0.218, 0.155); on emails – for AMT participants (0.085, -0.062); regular participants (0.330, 0.106) and overall (0.039, -0.052).

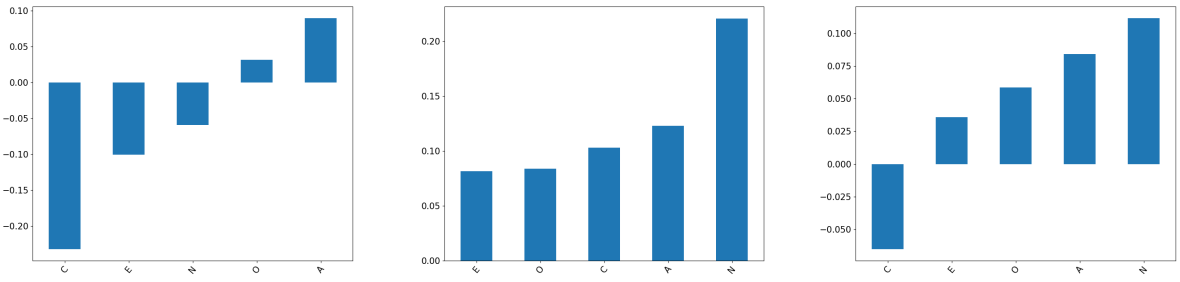
The regression analysis between the confidence levels and the human personality traits for the stories and emails have been done in a similar fashion and the regression correlation coefficients for the confidence level on detecting stories with human personality traits have been shown in Figure 64 with the regression equation shown in Equation 14. The same for emails are shown in Figure 65 and Equation 14. The $R^2$ and adjusted $R^2$ scores for the confidence levels on stories – for AMT participants (0.310, 0.138); regular participants (0.156, -0.226) and overall (0.078, -0.047); on emails – for AMT participants (0.233, 0.042); regular participants (0.345, 0.047) and overall (0.131, 0.013).

$$
\begin{aligned}
DR_{All,story} &= 0.403 * N + -0.327 * E + 0.036 * C + -0.018 * O + 0.017 * A \\
DR_{AMT,story} &= -0.374 * E + 0.369 * N + 0.068 * C + 0.011 * A + -0.005 * O \\
DR_{Reg,story} &= 0.314 * N + -0.149 * E + -0.067 * O + 0.046 * C + -0.006 * A
\end{aligned}
\tag{11}
$$




(a) (b) (c)

Figure 61: Comparing Regression Analysis Coefficients for human personality traits with Detection Rate on stories for (a) AMT (b) Regular (c) Overall.

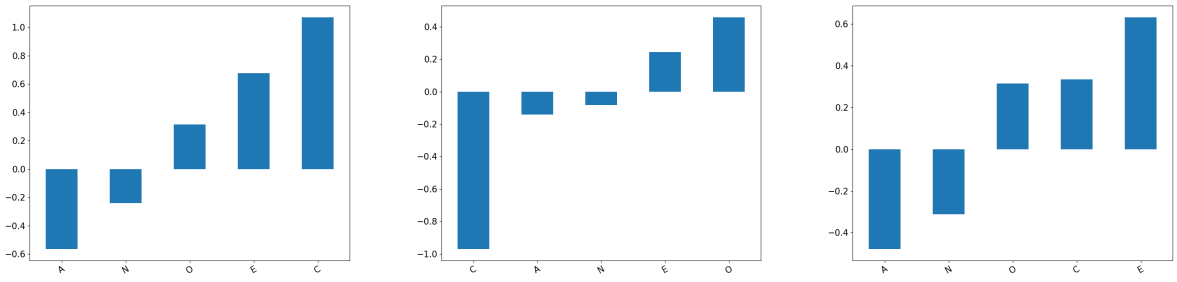



(a) (b) (c)

Figure 62: Comparing Regression Analysis Coefficients for human personality traits with Detection Rate on emails for (a) AMT (b) Regular (c) Overall.

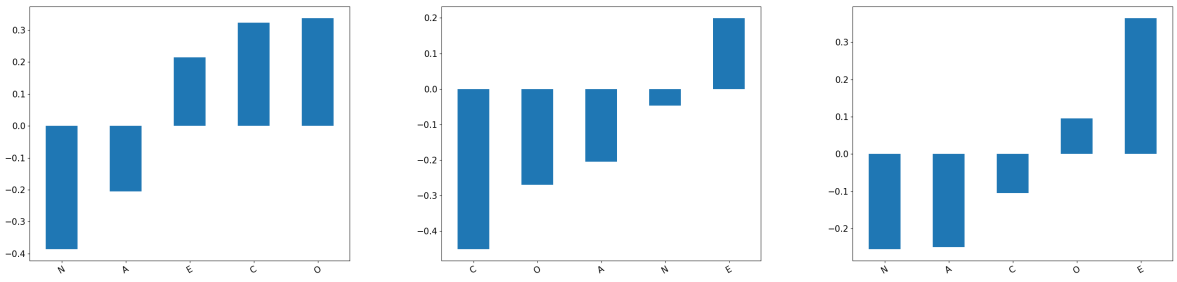
$$
\begin{aligned}
DR_{All,email} &= 0.112 * N + 0.084 * A + -0.065 * C + 0.059 * O + 0.036 * E \\
DR_{AMT,email} &= -0.232 * C + -0.101 * E + 0.09 * A + -0.059 * N + 0.031 * O \\
DR_{Reg,email} &= 0.22 * N + 0.123 * A + 0.103 * C + 0.084 * O + 0.082 * E
\end{aligned} \tag{12}
$$

$$
\begin{aligned}
CL_{All} &= -0.386 * N + 0.338 * O + 0.324 * C + 0.215 * E + -0.205 * A \\
CL_{AMT} &= -0.451 * C + -0.269 * O + -0.205 * A + 0.199 * E + -0.047 * N \\
CL_{Reg} &= 0.365 * E + -0.255 * N + -0.25 * A + -0.104 * C + 0.096 * O
\end{aligned} \tag{13}
$$



(a) (b) (c)

Figure 63: Comparing Regression Analysis Coefficients for human personality traits with Confidence Level on emails and stories for (a) AMT (b) Regular (c) Overall.

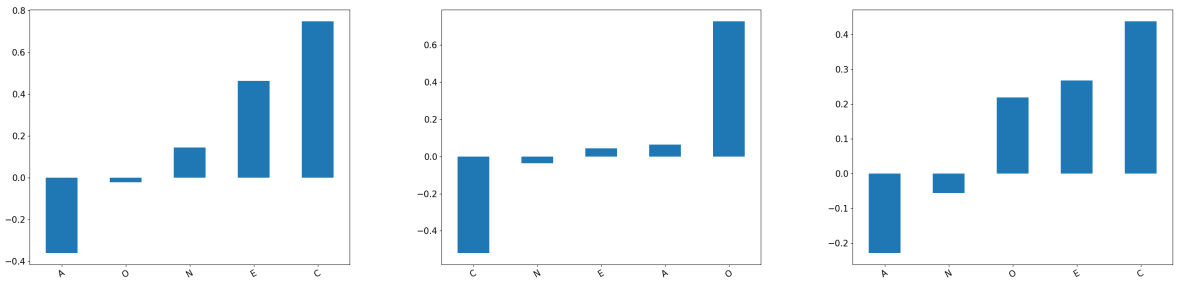


(a) (b) (c)

Figure 64: Comparing Regression Analysis Coefficients for human personality traits with Confidence Level on stories for (a) AMT (b) Regular (c) Overall.

$$
\begin{aligned}
CL_{All,story} &= 0.633 * E + -0.479 * A + 0.334 * C + 0.316 * O + -0.312 * N \\
CL_{AMT,story} &= 1.071 * C + 0.678 * E + -0.565 * A + 0.316 * O + -0.242 * N \\
CL_{Reg,story} &= -0.97 * C + 0.459 * O + 0.244 * E + -0.142 * A + -0.083 * N
\end{aligned} \tag{14}
$$

$$
\begin{aligned}
CL_{All,email} &= 0.439 * C + 0.268 * E + (-0.229) * A + 0.22 * O + (-0.057) * N \\
CL_{AMT,email} &= 0.748 * C + 0.463 * E + (-0.36) * A + 0.145 * N + (-0.022) * O \\
CL_{Reg,email} &= 0.728 * O + (-0.52) * C + 0.063 * A + 0.045 * E + (-0.036) * N
\end{aligned} \tag{15}
$$



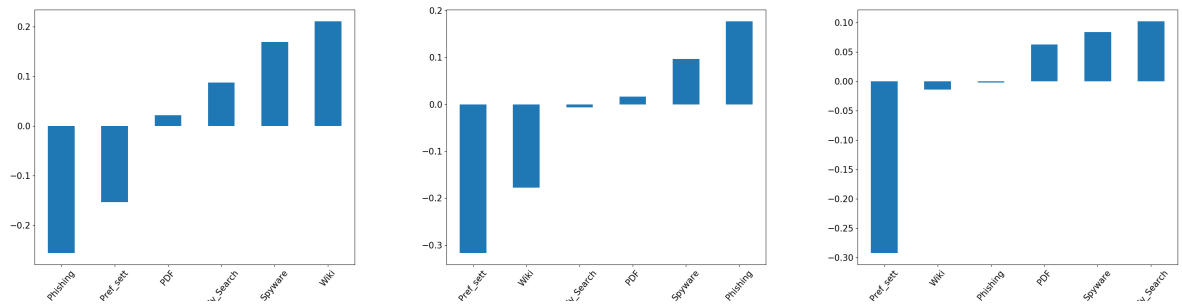

(a) (b) (c)

Figure 65: Comparing Regression Analysis Coefficients for human personality traits with Confidence Level on emails for (a) AMT (b) Regular (c) Overall.

### 7.4.2 Correlation with web usage skills

Similar to human personality traits' analysis, we analyze the web usage (wu) skills with the detection rate and confidence levels. Using the Ridge regression analysis, we first calculate the relationship between the web usage skills and the detection rate on both emails and stories of the participants by group and overall. The regression coefficients have been shown in Figure 66 for the AMT and Regular groups as well as overall. The quantitative values and their relation with the DR have been shown in the Equation 16 for the same. We observe the models' goodness of fit by calculating the $R^2$ and adjusted $R^2$ scores with the web usage – for AMT participants (0.167, 0.045); regular participants (0.153, -0.049) and overall (0.091, 0.016).

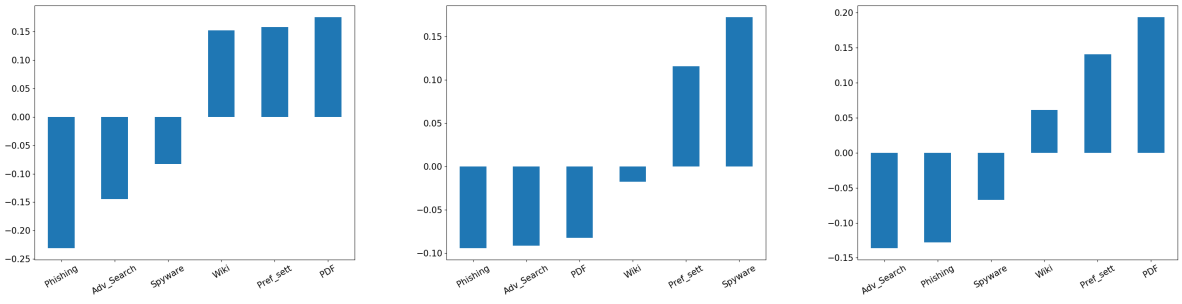
Figure 67 shows the correlation coefficients for the detection rate on stories with the web usage skills by group, the Equation 17 gives the equation for the regression model for stories. The same is shown for emails in Figure 68 shows the correlation coefficients for the detection rate with the web usage skills by group, the Equation 18 gives the equation for the regression model on emails. We report the $R^2$ and adjusted $R^2$ scores with the web usage for detection rate on stories – for AMT participants (0.180, 0.027); regular participants (0.102, -0.133) and overall (0.081, 0.008). The $R^2$ and adjusted $R^2$ scores for emails – for AMT participants (0.122, -0.054); regular participants (0.343, 0.062) and overall (0.071, -0.038).




(a) (b) (c)

Figure 66: Comparing Regression Analysis Coefficients for web usage skills with Detection Rate on emails and stories for (a) AMT (b) Regular (c) Overall

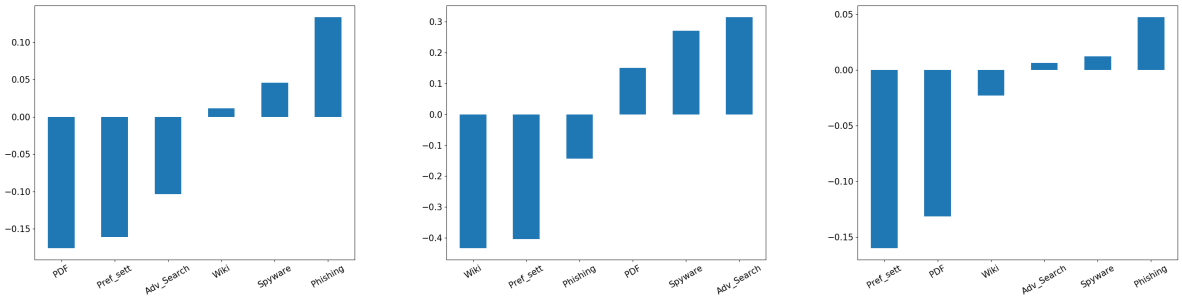
$$
\begin{aligned}
DR_{All} &= (-0.292) * Pref\_sett + 0.102 * Adv\_Search + 0.084 * Spyware + 0.063 * PDF \\
&+(-0.014) * Wiki + (-0.002) * Phishing \\
DR_{AMT} &= (-0.256) * Phishing + 0.21 * Wiki + 0.169 * Spyware + (-0.153) * Pref\_sett \\
&+0.087 * Adv\_Search + 0.022 * PDF \\
DR_{Reg} &= (-0.317) * Pref\_sett + (-0.178) * Wiki + 0.177 * Phishing + 0.096 * Spyware \\
&+0.017 * PDF + (-0.007) * Adv\_Search
\end{aligned}
\tag{16}
$$




(a) (b) (c)

Figure 67: Comparing Regression Analysis Coefficients for web usage skills with Detection Rate on stories for (a) AMT (b) Regular (c) Overall

$$DR_{All,story} = -0.231 * Phishing + 0.176 * PDF + 0.159 * Pref\_sett + 0.152 * Wiki$$
$$+ - 0.145 * Adv\_Search + -0.083 * Spyware$$
$$DR_{AMT,story} = 0.172 * Spyware + 0.116 * Pref\_sett + -0.094 * Phishing + -0.091 * Adv\_Search$$
$$+ - 0.082 * PDF + -0.017 * Wiki$$
$$DR_{Reg,story} = 0.194 * PDF + 0.141 * Pref\_sett + -0.136 * Adv\_Search + -0.128 * Phishing$$
$$+ - 0.067 * Spyware + 0.061 * Wiki \quad (17)$$



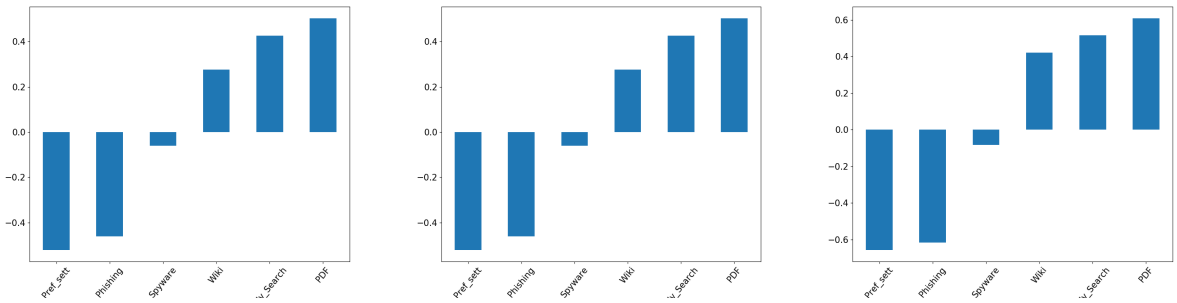

(a) (b) (c)

Figure 68: Comparing Regression Analysis Coefficients for web usage skills with Detection Rate on emails for (a) AMT (b) Regular (c) Overall

$$DR_{All,email} = -0.176 * PDF + -0.161 * Pref\_sett + 0.134 * Phishing + -0.104 * Adv\_Search$$
$$+0.046 * Spyware + 0.011 * Wiki$$
$$DR_{AMT,email} = -0.433 * Wiki + -0.404 * Pref\_sett + 0.315 * Adv\_Search + 0.271 * Spyware$$
$$+0.151 * PDF + -0.143 * Phishing$$
$$DR_{Reg,email} = -0.16 * Pref\_sett + -0.131 * PDF + 0.048 * Phishing + -0.023 * Wiki$$
$$+0.012 * Spyware + 0.007 * Adv\_Search \quad (18)$$

Our second analysis studies correlation between the **confidence level** and the web usage skills. The analysis algorithm is similar to that with detection rate (DR) as done previously. The correlation coefficients are shown in Figure 69 for the correlations with AMT and regular participants as well as correlation with the overall population. The equations in 19 show quantitatively the relation between the target variable, CL with the different web usage scores of the participant groups. The $R^2$ and adjusted $R^2$ scores with the web usage – for AMT participants (0.139, -0.132), regular participants (0.549, 0.279), and overall (0.163, 0.024).

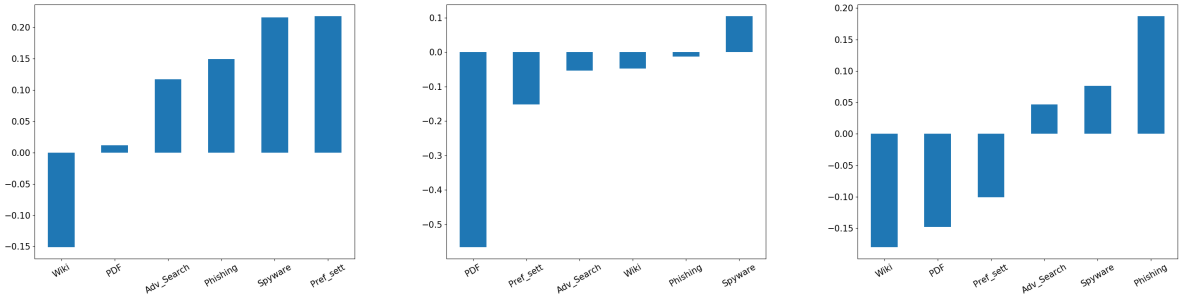
A regression analysis between the confidence level on detection of stories with the web usage skills for the different groups of participants has been shown in Figure 70 and the model equations have been shown in Equation 20. The same for emails are shown in Figure 71 and Equation 21. We analyze the models' goodness of fit by observing the $R^2$ and adjusted $R^2$ scores with the web usage – for stories, with AMT participants (0.115, -0.164), regular participants (0.272, -0.163), and overall (0.039, -0.121); for emails, with AMT participants (0.209, -0.041), regular participants (0.432, 0.091), and overall (0.191, 0.056).




(a) (b) (c)

Figure 69: Comparing Regression Analysis Coefficients for web usage skills with Confidence Level on both emails and stories for (a) AMT (b) Regular (c) Overall

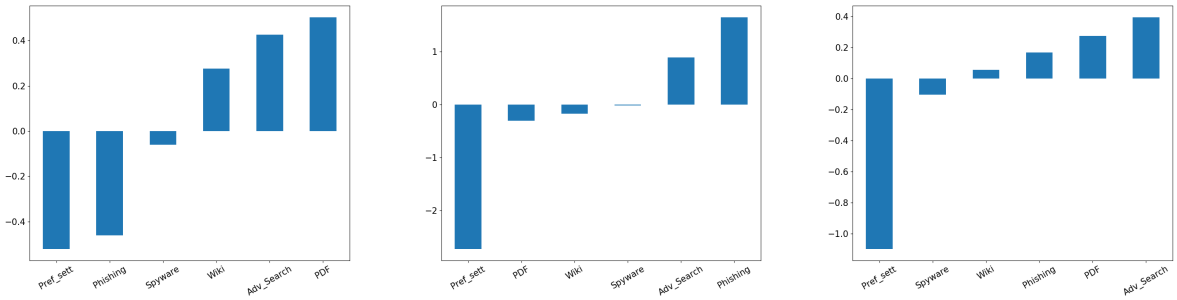
$$
\begin{aligned}
CL_{All} &= -1.199 * Pref\_sett + 0.442 * Adv\_Search + 0.356 * Phishing + 0.128 * PDF \\
&+ - 0.122 * Wiki + -0.029 * Spyware \\
CL_{AMT} &= 0.543 * Adv\_Search + 0.514 * PDF + -0.311 * Phishing + -0.304 * Pref\_sett \\
&+0.156 * Spyware + 0.126 * Wiki \\
CL_{Reg} &= -2.756 * Pref\_sett + -1.565 * PDF + 1.444 * Phishing + 0.885 * Adv\_Search \\
&+0.212 * Spyware + 0.183 * Wiki
\end{aligned}
\tag{19}
$$




(a) (b) (c)

Figure 70: Comparing Regression Analysis Coefficients for web usage skills with Confidence Level on stories for (a) AMT (b) Regular (c) Overall

$$
\begin{aligned}
CL_{All,story} &= 0.187 * Phishing + -0.18 * Wiki + -0.148 * PDF + -0.101 * Pref\_sett \\
&+0.076 * Spyware + 0.047 * Adv\_Search \\
CL_{AMT,story} &= 0.218 * Pref\_sett + 0.216 * Spyware + -0.151 * Wiki + 0.15 * Phishing \\
&+0.117 * Adv\_Search + 0.011 * PDF \\
CL_{Reg,story} &= -0.566 * PDF + -0.152 * Pref\_sett + 0.104 * Spyware + -0.054 * Adv\_Search \\
&+ - 0.048 * Wiki + -0.013 * Phishing
\end{aligned}
\tag{20}
$$




(a) (b) (c)

Figure 71: Comparing Regression Analysis Coefficients for web usage skills with Confidence Level on emails for (a) AMT (b) Regular (c) Overall

$$
\begin{aligned}
CL_{All,email} &= -1.099 * Pref\_sett + 0.395 * Adv\_Search + 0.275 * PDF \\
&+0.169 * Phishing + -0.105 * Spyware + 0.058 * Wiki \\
CL_{AMT,email} &= -0.521 * Pref\_sett + 0.503 * PDF + -0.46 * Phishing \\
&+0.425 * Adv\_Search + 0.277 * Wiki + -0.06 * Spyware \\
CL_{Reg,email} &= -2.721 * Pref\_sett + 1.644 * Phishing + 0.89 * Adv\_Search \\
&+ - 0.306 * PDF + -0.169 * Wiki + -0.017 * Spyware
\end{aligned}
\tag{21}
$$

## 7.5 Correlation of Automatic Metrics with Human Scores

We present an analysis of automated metric scores with human judgement on fluency and syntax. Using the stories and emails provided to the participants, we perform a Pearson's correlation ($\rho$) of the syntax and fluency scores on the graded textual instances. We observe a very high correlation of 0.968 in case of stories and a moderate $\rho$ of 0.88 in case of emails. To evaluate the correlation between the top automated metric with the human scores of fluency and syntax, we use the Point Biserial correlation analysis. The test is similar to Pearson's Correlation Coefficient but well-suited to analyzing correlation between a categorical and continuous variable. Table 35 shows the correlation coefficient between each automated metric with the fluency and syntax scores as graded

by the human participants for stories.[54] We separately show the correlation of the scores as done by the AMT and the Regular groups of participants. The same for the emails has been shown in Table 36.

Table 35: Correlation of automated metrics with human judgement scores on stories

| | $HM_{SC}$ | SC | $L_{SD}$ | $L_{avg}$ | TTR | BiOL | FRE | DCR | Noun | Verb |
|---|---|---|---|---|---|---|---|---|---|---|
| **Fluency** | 0.330 | -0.126 | -0.712* | -0.111 | -0.030 | 0.453 | 0.003 | 0.211 | 0.016 | 0.087 |
| **Fluency-AMT** | -0.109 | -0.112 | -0.745* | 0.314 | -0.291 | 0.085 | -0.273 | 0.124 | 0.018 | -0.153 |
| **Fluency-Reg** | 0.507 | 0.512 | -0.528* | -0.331 | 0.128 | 0.562 | 0.164 | 0.214 | 0.012 | 0.206 |
| **Syntax** | 0.238 | 0.100 | -0.699* | -0.174 | -0.022 | 0.420 | -0.002 | 0.273 | -0.098 | 0.152 |
| **Syntax-AMT** | -0.245 | -0.252 | -0.641* | 0.115 | -0.438 | -0.027 | -0.293 | 0.287 | -0.421 | 0.157 |
| **Syntax-Reg** | 0.443 | 0.447 | -0.603* | -0.291 | 0.195 | 0.574 | 0.148 | 0.218 | 0.084 | 0.123 |

Table 36: Correlation of automated metrics with human judgement scores on emails

| | $HM_{SC}$ | SC | $L_{SD}$ | $L_{avg}$ | TTR | FRE | DCR | Noun | Verb |
|---|---|---|---|---|---|---|---|---|---|
| **Fluency** | 0.045 | 0.048 | -0.200 | 0.033 | -0.082 | 0.097 | 0.160 | -0.018 | 0.154 |
| **Fluency-AMT** | 0.141 | 0.142 | -0.179 | -0.031 | -0.081 | -0.114 | 0.283 | 0.331 | -0.111 |
| **Fluency-Reg** | 0.007 | 0.009 | -0.167 | 0.046 | -0.067 | 0.142 | 0.091 | -0.116 | 0.201 |
| **Syntax** | -0.023 | -0.022 | -0.016 | 0.406 | -0.075 | 0.242 | 0.019 | 0.295 | 0.196 |
| **Syntax-AMT** | -0.083 | -0.084 | -0.226 | 0.019 | 0.028 | 0.143 | -0.140 | 0.214 | 0.023 |
| **Syntax-Reg** | 0.006 | 0.006 | 0.067 | 0.437 | -0.093 | 0.211 | 0.074 | 0.241 | 0.205 |

In Table 35 we see that for stories a **statistically significant yet negative correlation exists between the $L_{SD}$ metric and the human scores on fluency and syntax**. This is a significant improvement over the results shown in [65]. However, for the other automated metrics, we observe no other statistically significant correlation with the human scores. A moderate positive correlation exists between the metrics HM_sent_Conn and Bigram overlap percentage (BiOL) with fluency and syntax. The correlation analysis for human judgements on emails in Table 36 shows low correlation for all metrics with human scores on fluency and syntax. All the correlation results in email analysis were not statistically significant based on the $p-$value.

Therefore, we can say that there exists *little to no correlation* between human judgement and automated metrics. Moreover, the nature of the relationship between the two depends largely on the genre of the text evaluated. A much better correlation was observed for generated stories

[54] '*' shows the statistically significant results based on $p$-value

than for emails between the humans and the automated metrics. We can also conclude that while humans mostly base their judgements on the linguistic quality of the text, for the automated metrics it is system-specific. This supports the conclusions drawn in prior research on generated content evaluation [65].

# 8 Comparing Models and Genres

In this dissertation, we provide an in-depth evaluation and analysis of different classes of generative models and architectures for the purpose of content generation in the form of emails and stories. We compare the generated stories and emails from multiple generative architectures – pre-transformer wRNN (Chapter 4), pre-trained and fine-tuned transformer-based massive language models (Chapter 5) – GPT, Transformer-XL and XLNet and finally the sentence-level proposed hierarchical generative architecture or **HiGen** (Chapter 6), by using a wide range of quantitative metrics as well as by performing qualitative analysis. We test the performance of each model by using it to generate two genres of text – stories from writing prompts and organizational emails from subject prompts. The primary datasets used for training and evaluating the emails are the WritingPrompts dataset [27] of story-prompt pairs and legitimate organizational emails from Enron [26] and Avocado [23].

However, the quantitative analysis of the samples generated by each setup have been performed separately for each architecture in the Sections 4.2, 5.2 and 6.4. In this section, we perform a two-pronged in-depth study of the generated content (stories and emails) and the generative models based on their performance on the quantitative metrics – (a) an inter-model comparison and (b) an inter-genre comparison.

## 8.1 Inter-Model Comparison

We compare the story and email samples generated by the different generative models based on their performance on a selected set of metrics. We present our findings for the different models for the story and emails below. In the table 37 below we include the top two metrics based on the absolute value of their regression correlation coefficient calculated using the regression analysis between the metrics and the nature of the textual content. We see that for emails, sentence connectedness metrics are clearly the preferred choices. For stories, with regression models that have higher $R^2$ values or better fitness, $n$-gram based overlap with the conditioning prompt are the

top performing metrics.

Table 37: Metric Ranking on Story and Email datasets.

| Genre | Type | wRNN | PT-FT | HiGen |
|---|---|---|---|---|
| Story | **Negative** | HM_sent_conn, $L_{avg}$ | UniOL, TriOL | TriOL, FRE |
| | **Positive** | TTR, Sent_conn | BiOL, $L_{SD}$ | UniOL, Verb |
| Email | **Negative** | Sent_conn, Verb | FRE, Sent_conn | HM_sent_conn, Verb |
| | **Positive** | DCR, HM_sent_conn | HM_sent_conn, $L_{SD}$ | DCR, Sent_conn |

### 8.1.1 Comparison for Stories

The generative language models[55] were trained using BPE-tokenized versions of the WRITINGPROMPTS dataset [27]. The performance across the different models are then evaluated by analyzing the scores of a selected set of quantitative metrics on a random subset of stories generated by each model.

The comparative analysis across models helps us understand if there is a considerable change in the metric scores with nature of the generative model and also determine the nature of the change. The quantitative analysis of the models wRNN, pre-trained and fine-tuned language models and the proposed HiGen model is shown in Sections 4.2, 5.2 and 6.4 respectively. We use the results of the metric analysis in each section to identify the top three metrics for comparing the nature of the generated stories for the comprehensive study. We see that $n-$gram overlap percentage is one of the top metric of choice while comparing the generated stories – here, we select $biOL$ or bigram overlap percentage scores as one metric. The $L_{avg}$ metric or the mean sentence length in the generated samples is selected as the next metric for inter-model performance comparison. The final metric selected is the $HM\_sent\_conn$ or Harmonic Mean of Sentence Connectedness measures which is a property of the inter-sentence coherence for the generated stories. For our evaluation, we randomly selected 50 story samples generated by the wRNN model at $\tau = 0.75$; 50 each from the samples generated at $\tau = 0.75, k = 250$ by the pre-trained Transformer-XL (TX) and the fine-tuned w10-3M language models. We also selected 50 samples each generated

[55] with the exception of pre-trained models

by w5-1M_SBERT and OG_SBERT language models to build the dataset for our comprehensive evaluation. By analyzing the statistical properties of the afore-mentioned quantitative metrics, we can compare the performance of the generative models with respect to the generated stories.

**Story-Prompt Relatedness.** Figure 72 shows the statistical properties of the Bigram overlap percentage scores among the selected samples depending on the nature of the generative model. The boxplots compare the mean *biOL* score of the generated samples with the mean overlap score observed in human-written samples. The samples generated by the wRNN model have almost zero bigram overlap with the prompt, showing that the model has poor performance with respect to prompt relatedness or responsiveness. The low variability in the scores of the samples generated by the fine-tuned w10-3M model as well as a mean score closer to the human level, shows that the fine-tuned model generates samples which condition better on the given prompt.

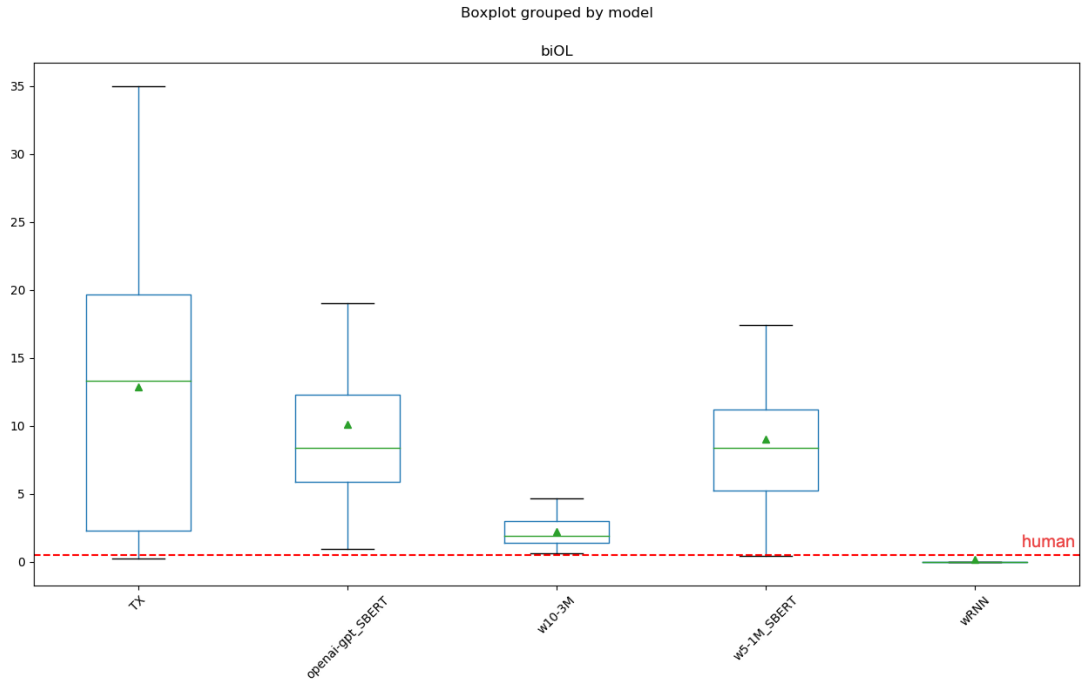



Figure 72: Comparing BiOL across best generative models for stories.

**Average Sentence Length.** Measuring the mean sentence length is a simple yet useful measure of syntactic style and complexity in textual content [79, 77]. In Figure 73 we compare the selected generative models by visualizing the statistical properties of the average sentence length

scores among the generated samples. The boxplots compare the mean $L_{avg}$ score of the generated samples with the mean of the human written sample stories. We observe that the mean of the average length of the samples generated by wRNN is statistically significantly similar to the mean human score. The same is observed for the samples generated by the fine-tuned w10-3M model.

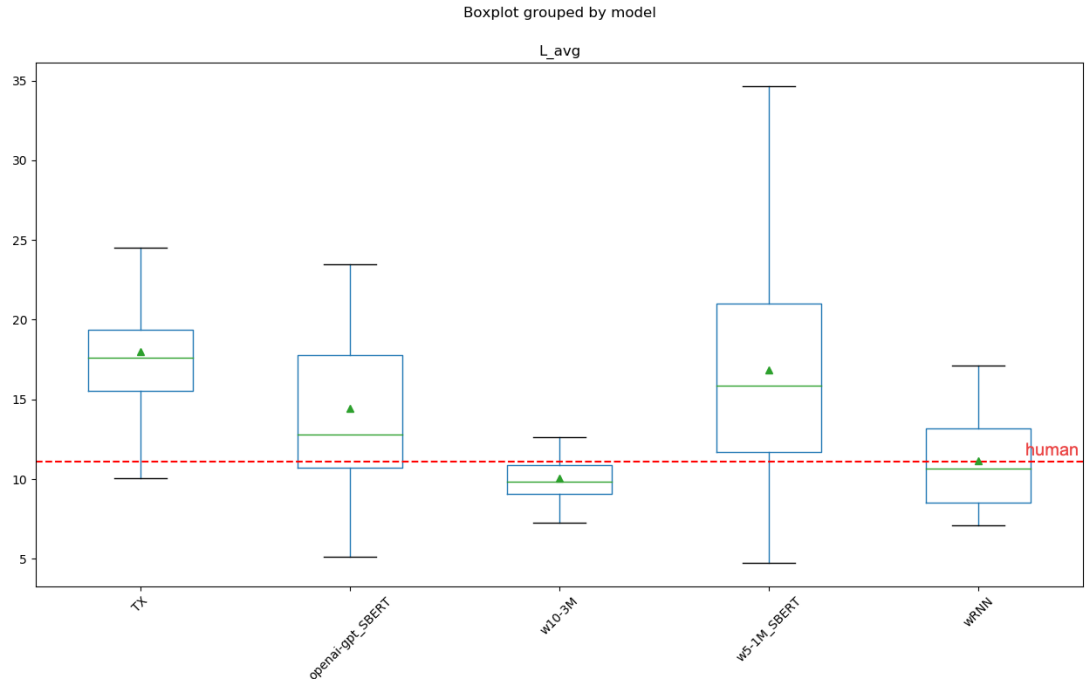



Figure 73: Comparing $L_{avg}$ across best generative models for stories.

**Harmonic Mean of Sentence Connectedness.** From our previous in-depth analyses, we have seen that the harmonic mean of the sentence connectedness' mean and standard deviation (HM_sent_conn) as a good measure of differentiating the nature of the textual content based on the type of author. Here, we take a closer look at the statistical properties of the metric calculated on random subsets of generated sample stories by different generative language models discussed previously. Figure 74 shows the variation in this metric across multiple generative models and compares the mean scores of the generated samples with the mean human baseline score. The mean scores of the samples generated by the HiGen models OG_SBERT and w5-1M_SBERT are shown to be statistically significantly closer to the mean score on human writing along with the samples from the pre-trained TX model.

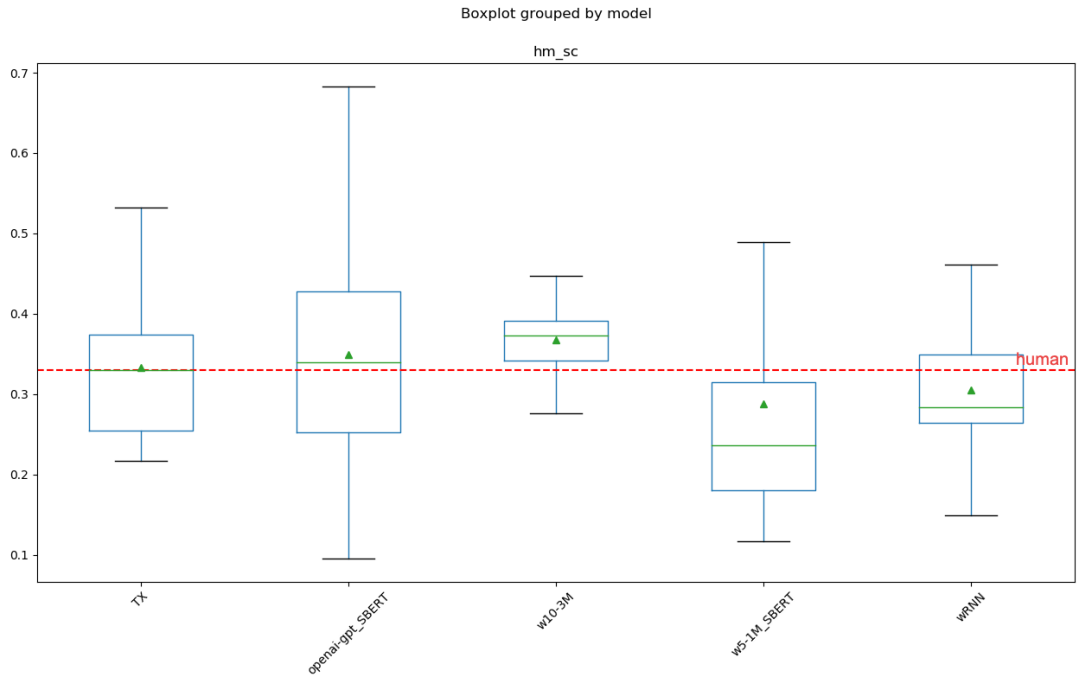



Figure 74: Comparing HM_sent_conn across best generative models for stories.

### 8.1.2 Comparison for Emails

Similar to the generated stories, we also performed an inter-model performance analysis of the generated emails. Studying the ranking of metrics for the three classes of generative model discussed in this research, we selected the following measures to study the change in model performance for email generation – Dale-Chall Readability scores (DCR), Harmonic Mean of Sentence Connectedness (HM_sent_conn) and Verb frequencies. We randomly selected 30 generated email instances from each of the following architectures and models – wRNN at $\tau = 0.75$, fine-tuned E5-3M at $\tau = 0.75, k = 50$, pre-trained OG at $\tau = 1.0, k = 10$, A5-3M_SBERT and TX_SBERT – to build our dataset for evaluation. We compared the model performance for the afore-mentioned metrics below.

**Email Readability.** The main purpose of an email is to convey the sender's intent to the receiver [78]. Thus, email readability is an important measure to compare the quality of the textual content generated in the form of an email body by the different generative models. Figure 75 shows the Dale-Chall Readability measure for the emails generated by the different models. We see that

while there exists a low statistical difference among the mean scores of the different models, they are statistically significantly greater than the mean human scores.

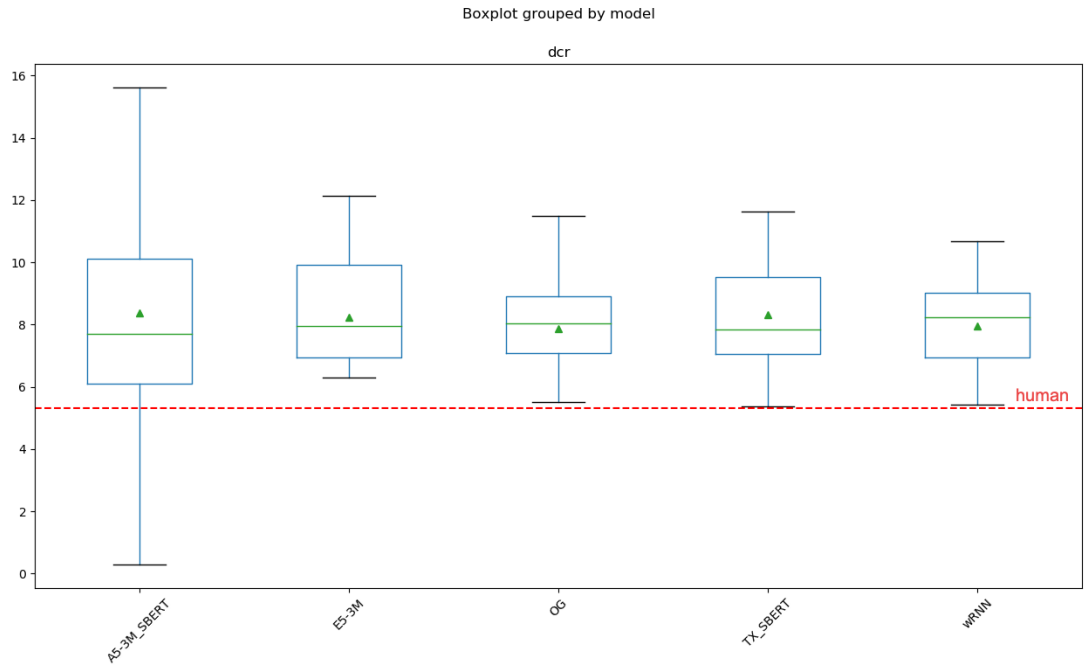



Figure 75: Comparing DCR across best generative models for emails.

**Harmonic Mean of Sentence Connectedness.** Similar to the evaluation on generated stories, we calculated the sentence connectedness metric in this setup. Figure 76 shows the statistical properties of the HM_sent_conn metric calculated on the samples generated by the different generative language models. We see that the mean $HM_{SC}$ score of the samples generated by the TX_SBERT model is significantly closer to the human levels.

**Verb Frequency.** The frequency of the POS tag Verb was shown to be a strong metric through the prior metric analysis studies. Figure 77 shows the statistical properties of the verb frequency in the samples generated by the different generative language models. We observe that while there exists no statistical similarity between the means of the measure on the generated samples with the human mean score, the mean verb frequency between the samples generated by the pre-trained OG and fine-tuned E5-3M are statistically significantly similar.

The above evaluation of the samples generated by the different models gives a comprehensive

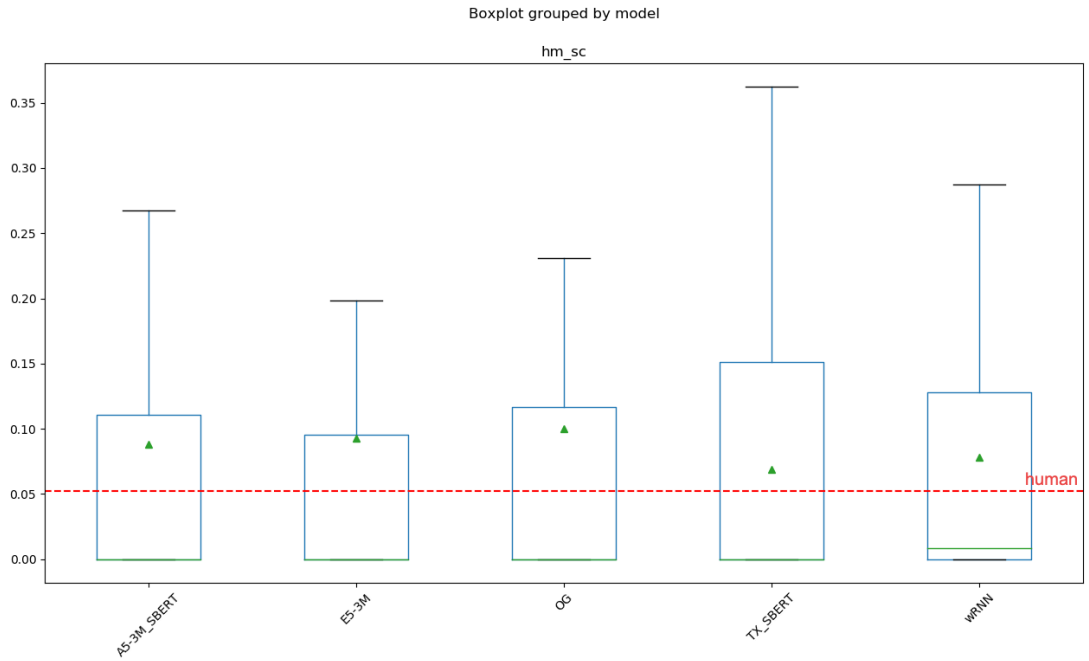



Figure 76: Comparing HM_sent_conn across best generative models for emails.

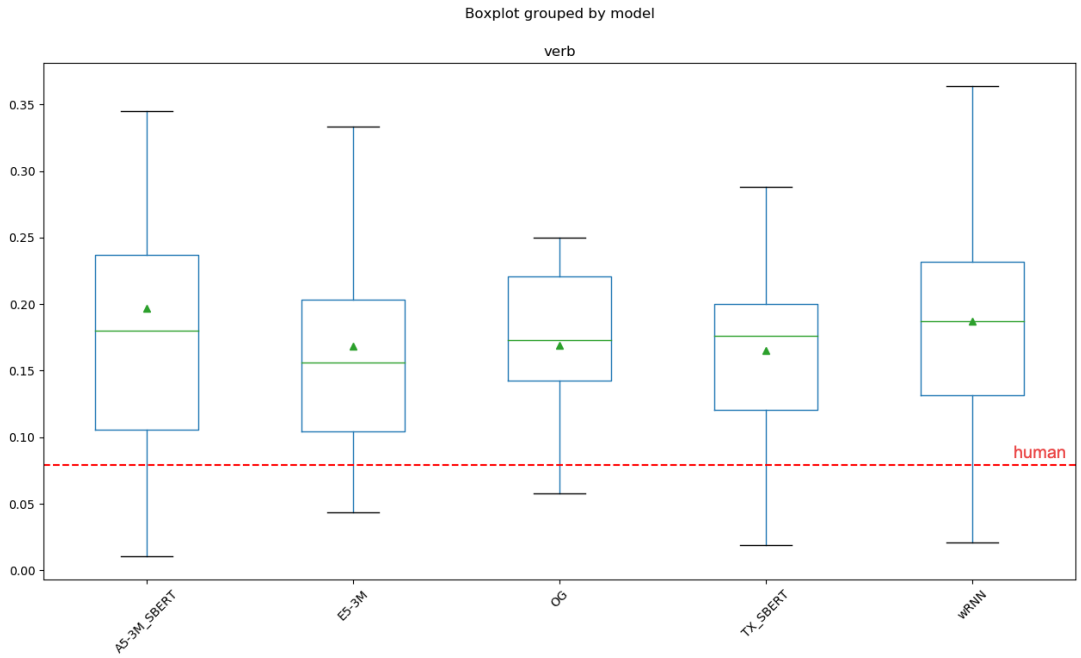



Figure 77: Comparing Verb frequency across best generative models for emails.

overview of the model performance with respect to the different metrics. It shows which models are performing better and closer to humans, also if some models are performing almost similar to

each other. Finally, this study can help us weigh the ranking of some models more than the others.

## 8.2 Inter-Genre Comparison

In this section, we evaluate the performance of metrics from a different point of view – the nature or genre of the data on which the models are trained. In prior analyses, the evaluation metrics used to score the samples generated by the trained and pre-trained language models were ranked based on their capability to distinguish the generated content from the human scores. We also look at the inter-metric correlation to evaluate whether there exists strong correlations among select set of measures.

Here, we review the inter-metric correlations for the different architectures with respect to the genre of the domain data. For the **wRNN architecture**, we observe that there exist strong correlations between the Sentence Connectedness metrics (*sent_conn* and $HM\_sent\_conn$) in Figures 14 and 15. Also, for the correlation values of the metrics' scores on the stories (Figure 14). Moderately high correlation exists among the TTR and DCR metrics and the average sentence length. The figure shows perfect correlation among the percentage of nouns and verbs. For emails, we do not observe much correlation between metrics in Figure 15. The correlation coefficients in the email correlation analysis are also much lower than those observed for the story samples. We also observe that the $R^2$ values of the metric ranking experiment for stories and emails are 0.92367 and 0.00257 respectively.

For the **transformer-based language models**, for stories there is a high positive correlation ($\rho > 0.9$) among the story-prompt $n-$gram overlap metrics (uniOL, biOL and triOL) as shown in Figure 26. Interestingly, these overlap metrics have relatively high positive correlation ($\rho > 0.75$) with the style metrics: average sentence length and the distribution of nouns in the textual content. Other interesting high positive correlations exist between the Dale-Chall Readability (DCR) scores and Type-Token Ratio, and between average sentence length and noun usage. For emails, Figure 27 shows that there exists no strong correlation among the metrics calculated on the generated emails. However, there exists a strong positive correlation between the sentence connectedness metrics

sent_conn and HM_sent_conn with $\rho = 0.98$.

For the different variations of the **HiGen architecture**, we analyzed the metric correlations using the mean scores of the different generative language models. Figure 39 shows that there exists a strong correlation among the overlap based metrics (uni-, bi- and tri-gram overlap) and between the readability metric DCR and type-token ratio. Also, there exists a moderate positive correlation between the length-based stylistic metrics (mean and std. dev. of sentence length) and $n-$gram based overlap. For emails, Figure 40 shows the low correlation that exists among the metrics – this can be attributed to the short text nature of the emails as well high sparsity in the email content. However, there exists a moderate correlation among the POS-tag based metrics (noun and verb) and the TTR percentage values. For the metric ranking analysis performed using the LassoCV algorithm, we observe – for the stories, the best model score (coefficient of determination, $R^2$) is 0.845068 and on the emails, the best $R^2$ is 0.010173.

# 9 Conclusions and Future Work

The research presented in this dissertation highlights the performance of the different generative language models in the realm of creative content generation as well as masqueraded fake email generation. We start our analysis with an in-depth overview of different deep neural architectures that have been leveraged for the purpose of text generation. This includes both pre-transformer simpler networks like Recurrent Neural Networks (or LSTMs) and transformer-based generative architectures. The effect of variation in architecture and sampling setups of deep neural learners have not been explored in-depth for text generation [79, 49].

We observe that the word-based generative language model built using Bi-LSTMs performs poorly during generation of both emails and stories. Using a generative massively trained language model greatly improves the quality of content generated, but with repetitions. The linguistic quality of the text generated by the system largely depends on the sampling setup and selection of hyperparameters. This choice of sampling parameters is a challenging, time-intensive task that is imperative for the selection of the best generative samples.

To address the selection of proper sampling paramaters during generation as well as for increased coherence at the sentence-level, we propose our hierarchical architecture – HiGen. Using a sentence level approach with improved sentence-level embeddings helps control the generation of coherent content from the given subject prompt. This becomes very important in case of email generation, since the intent and nature of the text generated becomes challenging to control. We show that by leveraging deep neural learners we can automatically generate emails emulating human writing style. These may deceive individuals and detection filters, tested by conducting a human task-based survey on email and story identification analysis. We start with word-based RNNs and apply the large pre-trained deep neural language models such as OpenAI's GPT and GPT2, Google/CMU's Transformer-XL and XLNet to open-ended story generation as well as email generation and test their generalizability. We also compare performance of the different language models across different genres of data using a wide variety of quantitative measures.

Using a variety of automated metrics that measure linguistic, syntactic and semantic quality of the generated stories, the language models are evaluated by comparing with human-written stories. Moreover, we also analyze the metrics with two techniques – a LASSO-based regression model and inter-metric correlation. In the exploratory analysis of metric importance, we see that the $n-$gram based overlap measures are the best performing followed by the standard deviation in sentence length (L_sd) for stories. For emails, the sentence connectedness metrics outperform the other measures and is followed by the readability metric scores (DCR and FKG).

Hence, we summarize the key findings of this research below:

- Transformer-based architectures outperform simpler neural architectures like RNNs in the task of text generation as seen on different genres of domain data – stories and emails. However, the performance of these models largely depend on the choice of sampling hyperparameters, softmax temperature and top-$k$ sampling value.

- We propose a hierarchical generative architecture, which takes into account the different sampling hyperparameter combinations while generating novel sentence candidates from conditioning prompts. The architecture generates the content one sentence at a time, comparing the generated sentence candidates to the sentence-embedding vectors selected by a predictive language model for choosing the best possible novel instance. This leads to a more textually coherent instance.

- We study the importance of the different evaluation metrics through a detailed regression based analysis. While metric ranking and goodness of fit is easier to explain for the task of story generation, the adjusted $R^2$ scores for the task of email generation are much lower making the evaluation of generated masqueraded emails a considerably difficult task.

- The task-based study with human participants show that fake emails are capable of deceiving humans successfully with minimal post-processing.

- Automated evaluation metrics are more effective for distinguishing generated stories, where the prompt overlap metrics perform the best. The proposed sentence connectedness based

metrics also have moderately high correlation with the human participant ratings. In absence of prompt overlap, sentence connectedness metrics perform the best for the task of effectively differentiating generated emails from human writing.

## 9.1 Future Work

While we perform an in-depth evaluation of the different generative systems starting from a baseline word-based recurrent neural network model to transformer-based language models. While the large generative language models have been used to generate textual content from a given prompt, the content is usually incoherent or repetitive in nature. We propose the HiGen architecture, which uses an additional predictive language model trained on sentence-level embeddings that acts as a guide to the generative architecture to select the best possible generated sentence candidates. Moreover, we propose sentence coherence measures – HM_sent_conn and Sent_conn which are moderately correlated with the human ratings. However, with respect to the application of these generative models, there still exists are number of possible future avenues that can be pursued. We present some of the areas of additional research that can be explored beyond the scope of this study:

- In the realm of creative content generation or writing stories, the generative models cannot properly condition on a prompt that is less of a story premise and more of an instruction. While human authors can identify such prompts, these action-based prompts usually tend to throw off a generative language model. We need to add additional attention-modeling layers or embeddings to provide the language model with information on how to process such conditioning prompts.

- For the purpose of this study, we only focus on open-ended content generation from given prompts, i.e., text-to-text modeling. In our future evaluation, we should extend the generative models' architecture and evaluate them on benchmark language generation datasets like E2E NLG dataset [56] (data-to-text generation), for the purpose of better model comparison.

[56] http://www.macs.hw.ac.uk/InteractionLab/E2E/

- Automatic generation of masqueraded emails is still a challenging task, as observed by our experimental setups and evaluation. Although we focus on training the improved transformer-based models only on legitimate or human-written emails from Enron and AvocadoIT corpora, an important aspect of such emails are their malicious cues. We therefore need to look into injection of malicious cues (some mentioned in Section 3.6) automatically into the generated emails.

- For the task of generation of emails, we post-process the generated content to remove some incoherent textual strings from the instance. For the purpose of this dissertation, we also focus only on one category – delivery of information for email generation. In future endeavours, we would like to explore how to model other intents automatically during generation.

- Automated evaluation of generated textual content is necessary to reduce the time and labor associated in human evaluation. Here, we evaluate the systems on generating two types of content – stories (narrative) and emails (conversational). The proposed metrics have been applied to the two genres, but in future we should focus on developing metrics that can differentiate between the two genres and can evaluate system performance separately for the two types [10, 101].

- We should also perform a close inspection to evaluate the existence of correlation between human judgement and the automated metrics presented in this work. This is important if we want to propose some evaluation metrics that can replace human bias and judgement effectively for evaluating generative text and models.

- The human evaluation study performed in this research should be extended to incorporate a much wider pool of participants. The control parameters of the study should also be improved so that the hypothesis can be appropriately formulated and analyzed. We should also improve upon the quality of the survey study by improving upon the questions and the rating scales (e.g., 7-point Likert scales) [3].

# Bibliography


[1] Ammar Almomani, T.-C Wan, A Manasrah, Altyeb Taha, M Baklizi, and S Ramadass. An enhanced online phishing e-mail detection framework based on evolving connectionist system. *International Journal of Innovative Computing Information and Control (IJICIC)*, 2012.

[2] Aman Rusia. XLNet speaks. Comparison with GPT-2. `https://medium.com/@amanrusia/xlnet-speaks-comparison-to-gpt-2-ea1a4e9ba39e`, 2019. Online; accessed 20 October 2019.

[3] Jacopo Amidei, Paul Piwek, and Alistair Willis. The use of rating and likert scales in natural language generation human evaluation tasks: A review and some recommendations. 2019.

[4] Madhusudhanan Chandrasekaran and Krishnan Narayanan and Shambhu Upadhyaya. Phishing email detection based on structural properties. In *NYS CyberSecurity Conf.*, 2006.

[5] Ram Basnet, Srinivas Mukkamala, and Andrew Sung. Detection of phishing attacks: A machine learning approach. *Soft Computing Applications in Industry*, pages 373–383, 2008.

[6] Alejandro Correa Bahnsen, Ivan Torroledo, David Camacho, and Sergio Villegas. Deepphish: Simulating malicious AI. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–8, 2018.

[7] Shahryar Baki, Rakesh Verma, Arjun Mukherjee, and Omprakash Gnawali. Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 469–482. ACM, 2017.

[8] Srinivas Bangalore, Owen Rambow, and Steve Whittaker. Evaluation metrics for generation. In *Proceedings of the First International Conference on Natural Language Generation-Volume 14*, pages 1–8. Association for Computational Linguistics, 2000.

[9] Shivam Bansal and Chaitanya Aggarwal. TextStat. `https://pypi.org/project/textstat/`, 2019. Online; accessed 20 October 2019.

[10] Kfir Bar, Vered Zilberstein, Ido Ziv, Heli Baram, Nachum Dershowitz, Samuel Itzikowitz, and Eiran Vadim Harel. Semantic characteristics of schizophrenic speech. *arXiv preprint arXiv:1904.07953*, 2019.

[11] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paass, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of Computer Security*, 18:7–35, 2010.

[12] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[13] Deanna D Caputo, Shari Lawrence Pfleeger, Joshua D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[14] Vitor R. Carvalho. *Modeling Intention in Email: Speech Acts, Information Leaks and Recommendation Models*. Springer Publishing Company, Incorporated, 2011.

[15] Ashwini Challah, Kartikeya Upasani, Anusha Balakrishnan, and Rajen Subba. Generate, filter, and rank: Grammaticality classification for production-ready nlg systems. *arXiv preprint arXiv:1904.03279*, 2019.

[16] Yun-Nung Chen and Alexander I Rudnicky. Two-stage stochastic email synthesizer. In *International Conference on Natural Language Generation*, pages 99–102, 2014.

[17] Yun-Nung Chen and Alexander I Rudnicky. Two-stage stochastic natural language generation for email synthesis by modeling sender style and topic structure. In *International Conference on Natural Language Generation*, pages 152–156, 2014.

[18] Savelie Cornegruta, Robert Bakewell, Samuel Withey, and Giovanni Montana. Modelling radiological language with bidirectional long short-term memory networks. *arXiv preprint arXiv:1609.08409*, 2016.

[19] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[20] A. Das and R. M. Verma. Can machines tell stories? A Comparative Study of Deep Neural Language Models and Metrics. *IEEE Access*, 8:181258–181292, 2020.

[21] Avisha Das and Rakesh Verma. Automated email generation for targeted attacks using natural language. In *Proceedings of the TA-COS Workshop at International Conference on Language Resources and Evaluation*, 2019.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. Avocado Research Email Collection. LDC2015T03. DVD. Philadelphia: Linguistic Data Consortium. (2015)., 2015.

[24] Christine E Drake, Jonathan J Oliver, and Eugene J Koontz. Anatomy of a phishing email. In *CEAS*, 2004.

[25] A. El Aassal, S. Baki, A. Das, and R. M. Verma. An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs. *IEEE Access*, 8:22170–22192, 2020.

[26] Enron Corpus. Enron Email Dataset. `https://www.cs.cmu.edu/\~enron/enron\_mail\_20150507.tar.gz`, 2015. Online; accessed 13 February 2018.

[27] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[28] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[29] Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. Topic-to-essay generation with neural networks. In *IJCAI*, pages 4078–4084, 2018.

[30] Ana Ferreira and Rui Chilro. What to phish in a subject? In *International Conference on Financial Cryptography and Data Security*, pages 597–609. Springer, 2017.

[31] Rudolf Franz Flesch. *How to write plain English: A book for lawyers and consumers*. Harpercollins, 1979.

[32] Daniil Gavrilov, Pavel Kalaidin, and Valentin Malykh. Self-attentive model for headline generation. In *European Conference on Information Retrieval*, pages 87–93. Springer, 2019.

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

[34] Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, 2016.

[35] Alberto Giaretta and Nicola Dragoni. Community targeted spam: A middle ground between general spam and spear phishing. *arXiv preprint arXiv:1708.07342*, 2017.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[37] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[38] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks : The Official Journal of The International Neural Network Society*, 18 5-6:602–10, 2005.

[39] James Grimaldi. U.S. Investigating Fake Comments on 'Net Neutrality'. `https://www.wsj.com/articles/u-s-investigating-fake-comments-on-net-neutrality-1544482280`, 2018. Online; accessed December 2018.

[40] Isabel Groves, Ye Tian, and Ioannis Douratsos. Treat the system like a human student: Automatic naturalness evaluation of generated text without reference texts. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 109–118, 2018.

[41] IsredzaRahmi A Hamid and Jemal H. Abawajy. Profiling phishing email based on clustering approach. In *Int'l Conf. on Trust, Security and Privacy in Computing and Communication*. IEEE, 2013.

[42] Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 44–52, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[43] David Hardcastle and Donia Scott. Can we evaluate the quality of generated text? In *LREC*. Citeseer, 2008.

[44] Eszter Hargittai and Yuli Patrick Hsieh. Succinct survey measures of web-use skills. *Social Science Computer Review*, 30(1):95–107, 2012.

[45] Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, 2014.

[46] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[47] Markus Huber, Stewart Kowalski, Marcus Nohlberg, and Simon Tjoa. Towards automating social engineering using social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 117–124. IEEE, 2009.

[48] HuggingFace Team. Write With Transformer. `https://transformer.huggingface.co/`, 2019. Online; accessed 20 October 2019.

[49] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Human and automatic detection of generated text. *arXiv preprint arXiv:1911.00650*, 2019.

[50] IT@Cornell. Phish Bowl. `https://it.cornell.edu/phish-bowl`, 2018. Online; accessed 10 February 2018.

[51] Glorianna Jagfeld, Sabrina Jenne, and Ngoc Thang Vu. Sequence-to-sequence models for data-to-text natural language generation: Word-vs. character-based processing and output diversity. *arXiv preprint arXiv:1810.04864*, 2018.

[52] Samaneh Karimi, Luis Moraes, Avisha Das, Azadeh Shakery, and Rakesh Verma. Citance-based retrieval and summarization using IR and machine learning. *Scientometrics*, 116(2):1331–1366, 2018.

[53] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, 2016.

[54] J Peter Kincaid, Robert P Fishburne Jr., Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and Flesch reading ease formula) for navy enlisted personnel. Technical report, Defense Technical Information Center (DTIC) Document, 1975.

[55] Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[56] Mirella Lapata and Regina Barzilay. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, volume 5, pages 1085–1090, 2005.

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

[58] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

[59] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[60] Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. Curriculum learning for natural answer generation. In *IJCAI*, pages 4223–4229, 2018.

[61] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[62] Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*, 2017.

[63] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of The International Speech Communication Association*, 2010.

[64] Marcus Nohlberg and Stewart Kowalski. The cycle of deception: A model of social engineering attacks, defenses and victims. In *Second International Symposium on Human Aspects of Information Security and Assurance (HAISA 2008), Plymouth, UK, 8-9 July 2008*, pages 1–11. University of Plymouth, 2008.

[65] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for NLG. *arXiv preprint arXiv:1707.06875*, 2017.

[66] OpenAI. Better Language Models and Their Implications. https://openai.com/blog/better-language-models/, 2019. Online; accessed 20 October 2019.

[67] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[68] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[69] Richard D Powers, William A Sumner, and Bryant E Kearl. A recalculation of four adult readability formulas. *Journal of Educational Psychology*, 49(2):99, 1958.

[70] Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

[71] Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. Assigning personality/profile to a chatting machine for coherent conversation generation. In *IJCAI*, pages 4279–4285, 2018.

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

[73] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[74] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*, 04 2020.

[75] Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009.

[76] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[77] Melissa Roemmele, Andrew S Gordon, and Reid Swanson. Evaluating story generation systems using automated linguistic analyses. In *SIGKDD 2017 Workshop on Machine Learning for Creativity*, pages 13–17, 2017.

[78] Maya Sappelli, Gabriella Pasi, Suzan Verberne, Maaike de Boer, and Wessel Kraaij. Assessing e-mail intent and tasks in e-mail messages. *Information Sciences*, 358:1–17, 2016.

[79] Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*, 2019.

[80] Sattar Seifollahi, Adil Bagirov, Robert Layton, and Iqbal Gondal. Optimization based clustering algorithms for authorship analysis of phishing emails. *Neural Processing Letters*, 46(2):411–425, Oct 2017.

[81] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[82] John Seymour and Philip Tully. Weaponizing data science for social engineering: Automated E2E spear phishing on twitter. *Black Hat USA*, page 37, 2016.

[83] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382. ACM, 2010.

[84] Jordan Shropshire. Natural language processing as a weapon. In *Proceedings of the 13th Pre-ICIS Workshop on Information Security and Privacy*, volume 1, 2018.

[85] Priya Sidhaye and Jackie Chi Kit Cheung. Indicative tweet generation: An extractive summarization problem? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 138–147, 2015.

[86] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316. IEEE, 2010.

[87] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*, 2017.

[88] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[89] Erik Stayton and Antonio Roque. Data-driven multi-agent email generators. In *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–7. IEEE, 2018.

[90] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[91] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[92] Robert Tibshirani. Regression shrinkage and selection via the lasso. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 58:267–288, 1994.

[93] The New York Times. Sarah Palin emails: The Alaska archive. `http://documents.latimes.com/sarah-palin-emails/`, 2011. Accessed: 2016-08-14.

[94] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, 2019.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[96] Ramanathan Venkatesh and Wechsler Harry. Phishing detection and impersonated entity discovery using conditional random field and latent dirichlet allocation. *Computers & Security*, 34:123–139, 2013.

[97] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. Detecting phishing emails the natural language way. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security – ESORICS 2012*, pages 824–841, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[98] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. Two-pronged phish snagging. In *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pages 174–179. IEEE, 2012.

[99] Rakesh M. Verma and Ayman El Aassal. Comprehensive method for detecting phishing emails using correlation-based analysis and user participation. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, 2017*, pages 155–157, 2017.

[100] Rakesh M. Verma and Nabil Hossain. Semantic feature selection for text with application to phishing email detection. In *Information Security and Cryptology – ICISC 2013*, pages 455–468, Seoul, Korea, 2014. Springer International Publishing.

[101] Christina Reuterskiöld Wagner, Ulrika Nettelbladt, Birgitta Sahlén, and Claes Nilholm. Conversation versus narration in pre-school children with language impairment. *International Journal of Language & Communication Disorders*, 35(1):83–93, 2000.

[102] Wei Wang, Saghar Hosseini, Ahmed Hassan Awadallah, Paul N Bennett, and Chris Quirk. Context-aware intent identification in email conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 585–594. ACM, 2019.

[103] Wikileaks. Hillary Clinton email archive. `https://wikileaks.org/clinton-emails/`, 2016. Accessed: 2016-08-14.

[104] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*, 2018.

[105] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[106] Stanley C Xie, Ruchir Rastogi, and Max Chang. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation. *Natural Language Processing with Deep Learning*, 2017.

[107] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[108] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. Automated crowdturfing attacks and defenses in online review systems. *arXiv preprint arXiv:1708.08151*, 2017.

[109] Denis Yarats and Mike Lewis. Hierarchical text generation and planning for strategic dialogue. *arXiv preprint arXiv:1712.05846*, 2017.

[110] Adwan Yasin and Abdelmunem Abuhasan. An intelligent classification model for phishing email detection. *CoRR*, abs/1608.02196, 2016.

[111] John Yearwood, Musa Mammadov, and Arunava Banerjee. Profiling phishing emails based on hyperlink information. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, 2010.

[112] Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Zonghan Yang. Chinese poetry generation with a working memory model. *arXiv preprint arXiv:1809.04306*, 2018.

[113] Gaoqing Yu, Wenqing Fan, Wei Huang, and Jing An. An explainable method of phishing emails generation and its application in machine learning. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1279–1283. IEEE, 2020.

[114] Weider D. Yu, Shruti Nargundkar, and Nagapriya Tiruthani. Phishcatch - a phishing detection tool. *2009 33rd Annual IEEE International Computer Software and Applications Conference*, 2:451–456, 2009.

[115] Masoumeh Zareapoor and Seeja K. R. Feature extraction or feature selection for text classification: A case study on phishing email detections. In *Engg. and Electronic Business*, 2015.

[116] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*, 2019.

[117] Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. Reinforcing coherence for sequence to sequence model in dialogue generation. In *IJCAI*, pages 4567–4573, 2018.

[118] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.