# How to Tell Real from Fake: research on text generation and linguistic feature in text classification

**Jiaqi Zhang**

A report submitted for the course
COMP4560 Advanced Computing Project
Supervised by: Patrik Haslum

The Australian National University

Draft — June 2019

June 2019

Except where otherwise indicated, this report is my own original work.

Jiaqi Zhang
1 June 2019

# Acknowledgments

Thanks for my supervisor, Dr Patrik Huslum in his support and instruction in the whole process of this project.

# Abstract

This paper has a look into the text generation task and classification task. The main focus and distinguish design in this paper is to introducing the linguistic features in to the classification task and tried to find the contribution of linguistic features of a text in the classification task. These findings can also be meaningful in making improvement in text generation tasks.

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

## 1.1 Problem Statement

With artificial intelligence techniques developing rapidly, 'computers' can do many things that were thought only could be done by humans, like 'writing text' by themselves. Text generation is the academic description for 'writing text by computer', which is an important part in the Natural Language Process area. Text generation has a great application prospect, playing a key role in machine translation [46], automatically question-answering system [47], summarization [48] and other tasks.

Owing to the large number of sources computers can learn from, they can imitate humans to write a wide range of text from a Twitter message to a novel, or continuous questioning and answering. For example, the automatic question and answering system ELIZA has been an important sign of the maturity of the application of text generation in NLP. The ELIZA system and other question-answering system have shown good performance in the Turing test. However, it is usually easy to tell the imitations of longer generated text from a real text written by human, like news and film reviews. People can tell the differences after only reading a bit of them. But it is not that easy for computers to tell the difference.

The point of this paper is to do some investigation in different approaches to make computers learn to tell the imitations by giving them a text and training them to decide whether it is written by a human or a computer. Compared with former researches on text classification tasks, we choose the computational linguistic features, which extracts the human understandable features into computational ones as machine algorithm input, as the primary features in classification instead of simply random extracted features. By doing that, we can potentially make the computer write better text, and try to figure out the thinking model of humans when reading and understanding a text.

## 1.2 Motivations

Language is the way human uses explain to themselves and communicate with each other, and plays the vital role as carrier of information. Text is the written expression of language. It is a consolidate and construct of information. By building advanced

text generation, we are building the ability of machines to arrange information and present it in an understandable way for humans. It can help human in many ways, like machine translation, speech summarization, and intelligent voice control system. Text generation can not only help people with a better communication by making it easier to express and understand and save labor costs, but also construct a bridge of communication between human and machine, which may break 'the fourth wall' in technique.

For better text generation, we want to get generated text has fewer differences from text written by human. This is making machine to have clearer expression from human aspect so helping human to have easier understanding of the 'machine written' text.

## 1.3 Project Scope

In this paper, we carry out research from text generating to generated text classification. The state of art technique for text generation is GAN, the Generative Adversarial Network. A GAN model is composed of two confrontational networks, the generator and the discriminator. The two networks work against each other, and make improvement according to the feedback until they reach a balance. Based on the GAN theory, the two networks actually promote each other. A better classification network can help to improve the generation of text.

For the text generation network, we mainly used Texygen, a benchmarking platform for text generation models. This platform gathers various generation models based on likelihood, such as SeqGAN, LeakGAN, RankGAN, and TextGAN. These models are all Generative Adversarial Networks, but each has different advanced designs. We used different models from Texygen platform for the research analyses of generation part.

## 1.4 Report Outline

In this report, we will start the discussing from different GAN models for text generation, and the performance comparation of different GAN models on different dataset. Then we will focus on extracting different linguistic features from text, mainly by different auxiliary word embedding tasks.

In both parts, we will introduce the state of art techniques in the area, followed by the experimental research on different approaches. By doing the qualitative and quantitative experiments, we tried to get a intuitive result on the comparation of these approaches and the influences of different linguistic features in text generation and generated text classification.

How many chapters you have? You may have Chapter 2, Chapter 3, Chapter 4, Chapter **??**, and Chapter 5.

Chapter 2

# Background and Related Work

## 2.1 Background

The original definition of a natural language generation system was to take non-verbal information as input and generate readable text expressions [20]. Generating data into text applies to this definition. Wan et al. extended this concept to include text-to-text generation, data-to-text generation, and image-to-text generation [21].

Text generation is an important research direction in the field of natural language processing. Its application has broad prospects. As the technology was development, it has penetrated every aspect of our life even before we have noticed. Automated Insights, Narrative Science companies and other text generation systems have been put into use in actual production. These systems generate other explanatory text, such as news, financial reports, and so on, from formatted data or natural language text. For example, organizations like the Associated Press use Automated Insights' WordSmith technology to cover college football, corporate earnings and more. The app not only allows the Associated Press to speed up news updates, but also expands coverage of the company's financial results while retaining its human resources [22].

## 2.2 Related work

### 2.2.1 Text generation

According to the data type of input, there are mainly three different types of text generation: data-to-text generation, text-to-text generation and image-to-text generation.

The data-to-text generation is used in the analysis or summarization task for structured data. The data-to-text generation system was divided into four parts, signal processing, data analyzing, document planning and text implementation by Reiter [23]. According to Mei and his research fellow [24], they have introduced aligner into encoder-decoder model to select important information, which is an end-to-end data-to-text generation model based on deep learning.

There are also different applications in image-to-text generation, like image caption, question answering and story generation based on image. The COCO dataset by Microsoft is a well known and constructed example for the application of image

caption. Kenneth and Francis [25] put forward the task to generating story according to image sequences, and provided the database of three levels, the description text for single image, stories for single image and stories for image sequence.

Text-to-text generation is the focus for our research. The main research fields in text-to-text generation are document summarization [26][27], sentence compression [28], paraphrase generation [29] and machine translation, which Google has published a machine translator network structure based entirely on attention [30] to achieve the new state of the art performance.

### 2.2.2 Generative Adversarial Net and its application in NLP

The Generative Adversarial Nets (GAN) [1] approach has come to prominence in the field of machine simulated data generation. GAN has shown a good performance in image generation [14]. NVIDIA has published the progressive growing of GANs for generating celebrity face image [2], which has a better result than previous work. And the generated images are sometimes hard to tell from real even by human.

GAN models are composed of two different parts, the generator model G and the discriminator D. The goal of generator is to generate data looks 'real', and discriminator is working to distinguish generated data from real data. The operating principle of the whole GAN model is a game between the generator and discriminator. The discriminator would classify real data the data generated by generator, then the generator would learn from the classification result and try to generate better data, which is harder for discriminator to identify. The game would last until the two models reach a balance and can not improve themselves by learning from each other.

The GAN model gives the state of the art result in image generation. But there is a problem when it comes to text generation. The generator and discriminator are constructed from differentiable functions. However, the text data has a discrete structure which is non differentiable, on which the simple GAN model cannot work. To solve this problem, researchers have created many variations of GAN for text data, like the SeqGAN, LeakGAN, RankGAN and other GAN models included in the Texygen platform.

The research on GAN's application in natural language processing also shows a growing trend, such as dialogue generation [8], text modeling [10][11][9], neural machine translation [12], question and answer [13], etc. However, in the NLP task, GAN is more difficult to train and requires extra modifying, which makes it an interesting but challenging research field.

### 2.2.3 Text classification

Text classification is an important research area in natural language processing. Unlike image data, text data has its special character of discrete and variable length. Thus, the traditional convolutional neural network which takes data of certain size as input and works well on image input does not have a good performance on text data.

To create network structures that are more suitable for discrete text data, researchers have come up with different modified models. FastText is a simple and efficient approach for text generation [32], and is always used as the baseline model. FastText works like the word2vec approach instead of a deep learning model, which leads to a great saving in training time, and can also show a good performance. Baoxin [31] combined CNN and RNN to develop the DRNN structure, which introduced the translation invariance into RNN and used CNN in the unit in RNN. Zeyang and Yujiu used the Multi-sentiment-resource Enhanced Attention Network (MEAN) to introduce attention into neural network [33]. The network constructed different lexicons for sentiment words, negative words and emphatic words for the sentiment classification tasks.

Chapter 3

# Methods used

## 3.1 GAN text generation

The Generative Adversarial Nets (GAN) approach has come to prominence in the field of machine simulated data generation, and mainly used in the field of computer vision and natural language processing. GAN approach has shown a good performance in the area of image generation, but the simple GAN does not work so well in the text generation. Thus, there are many modified GAN models to satisfy the text generation problem. The Texygen platform [3] includes the most commonly used GAN models, which shows state of the art or nearly state of the art result. Among these models, we have found that the SeqGAN [4], RankGAN [5], TextGAN (feature matching) [6] and LeakGAN [7] show better results. The rest of this section gives a brief introduction to these models.

### 3.1.1 SeqGAN

The biggest limit of using GAN models for text data generation is the non-differentiability of discrete data. To solve this problem, the Sequence Generative Adversarial Nets (SeqGAN) combines reinforcement learning with GAN model.

As shown in the figure above (left), still the idea of adversarial, real data and the

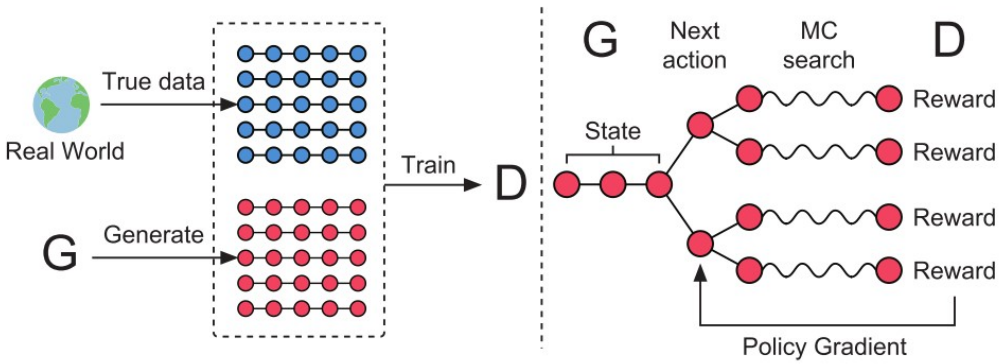



**Figure 3.1:** The structure of SeqGAN model

generated data from G are combined to train D. But as the problem said before, we know that with the discrete output of G, it is very difficult for D to back propagate the gradient used to update G. Thus, as the right part of the figure shows, the paper takes the policy network as G. There is a red dot called the state of now (state), and the next red dot to generate is called action (action). Because a complete sequence is needed for D to score, MCTS (Monte Carlo Tree Search) is used to complete the possibilities of every movement. D will produce a reward for these complete sequences and send them back to G to update G through reinforcement learning. This is to use Reinforcement learning to train a generation network which can produce the next optimal action.

### 3.1.2 RankGAN

Rank Generative Adversarial Nets (RankGAN) modified the discriminator of SeqGAN to get language description of high quality. The traditional discriminator in GAN is a binary classification model, which is only used to determine whether the input data comes from the real distribution or the generator. However, in RankGAN, the discriminator is seen as an optimization of learning to rank, turns the discriminator into ranker, which makes the discriminator rank lower for generated data than for real data. Thus, the reward in RankGAN contains the relative ordering information.

### 3.1.3 TextGAN

TextGAN was also proposed to solve the problem of discrete output in GAN model. But unlike the SeqGAN, TextGAN uses LSTM as the generator, and CNN as discriminator for GAN [34]. The idea of smooth approximation was introduced for the output of the LSTM generator to solve the problem of non-differentiable gradient caused by discretization. Authors also used multi training methods to improve the convergence of the GAN model, as well as the approach of feature matching for the optimization function.

The adversarial feature matching is an improvement to the objective function of the original GAN. Kernelized discrepancy metric is used to match the hidden feature representation of real sentences and generated sentences, which is efficient for relieving the mode collapse in adversarial training.

### 3.1.4 LeakGAN

For the methods before, the reward in GAN model would only be returned after complete sequence is generated by the generator. In the process of sequence generation, there is a lack of feedback on the intermediate information of sequence structure, and the reward scalar does not have much information about the semantic structure of sequence, so it cannot play a special role in guiding, which limits the length of sequence generation (generally no more than 20).

The LeakGAN was proposed to solve this problem of long text generation. The discriminator will leak some extracted features to the generator in the middle time

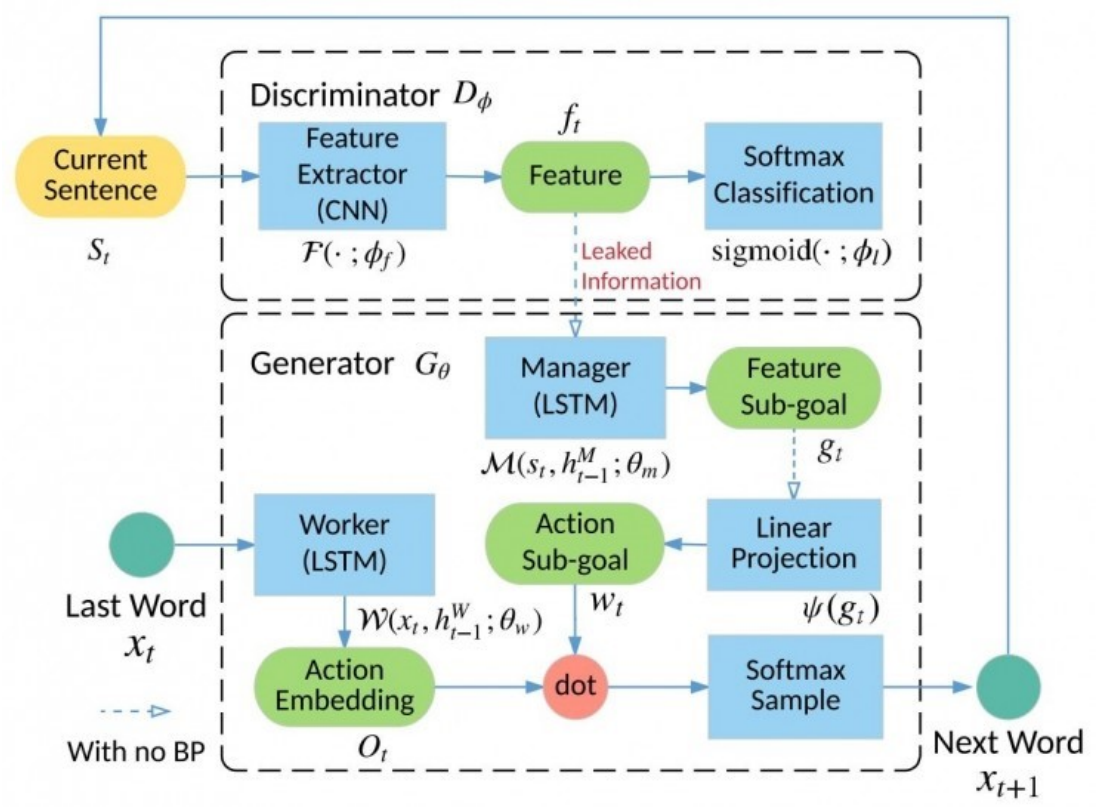



**Figure 3.2:** The structure of LeakGAN model

step, and the generator will use this additional information to guide the generation of the sequence.

As shown in the feature above, the generator uses hierarchical reinforcement learning structure, including the Manager and Worker module. The Manager module is an LSTM network that acts as a mediator. At each time step, it will receive a feature representation (for example, feature map in CNN) from the discriminator, and then pass it as a guidance signal to the Worker module. Because the intermediate information of the discriminator should not be known by the generator in the original GAN, the characteristic representation is called the leaked information. After receiving the embedding of this guidance signal, the Worker modules are also using LSTM network coding as current input. Then the output of the LSTM embedding is combined with the received guidance signal to calculate the next action, which is to choose the next word.

## 3.2 Generated text classification

Text classification has been an elementary task in natural language processing researches. There are many models that have shown the state of the art results, like textCNN [35], DCNN [36], RCNN [37], and HAN [38]. However, different from

```
The sequel, Yes, Prime Minister, ran from 1986 to 1988.
In total there were 38 episodes, of which all but one lasted half an hour.
Almost all episodes ended with a variation of the title of the series spoken as the answer to a question posed by the same character, Jim
Hacker.
Several episodes were adapted for BBC Radio, and a stage play was produced in 2010, the latter leading to a new television series on UKTV
Gold in 2013.
```

**Figure 3.3:** A sample text contains 4 sentences. The system has separate the sentences and will find the entities for each one.

previous research, our classification task is focused on the linguistic features in text, to see how these features contribute to the classification of human written text and machine generated text. In our experiment, we have mainly explored three linguistic aspects for the text: coherence, selectional preference and subject analysis by text summarization.

### 3.2.1 Coherence

Coherence is an important feature for humans to understand a text, which attracts considerable attention of the coherence feature in text generation tasks. Various theories about coherence have been developed over years of research [39][40]. However, the previous work on the coherence feature either depend a lot on handcrafted rules, which cause limitation in the application domain, or are represented in a complex embedding form, which makes it hard to implement in a robust system. Thus, here we implemented an entity-based approach based to local coherence by Barzilay and Lapata [41].

The approach focuses on local coherence which captures text associations at the conversion level between sentences. It provides a way to represent the text by a two dimensional grid of entities which captures the distribution of discourse entities in textual sentences. The entity here refers to a set of noun phrases having the same referent. Table 2 shows some examples of the entities in sentences given in the paper of Barzilay and Lapata.

In this presentation, we consider not only the appearance of the entities, but also their grammatical roles as subjects (s), objects (o), or neither (x). These grammatical roles are extracted using the robust Collins's [42] statistical parser. One of the core research problems in building entity-based coherence models is to determine which sources of linguistic knowledge are important for accurate prediction and how to encode them succinctly in textual representations. The present study largely agrees on the choice of entity distribution characteristics related to local coherence, but the difference is in the way these characteristics are modeled [41].

Information and grammatical roles of entities in the discourse are represented in the form of an entity grid like the table 1 shows:

As we can see in table 1, the six rows in the table indicates the number of sentences in the given discourse. The word for each column represents an entity, and the sign in the table shows the presence and grammatical role of the entity in each sentence, where S means the entity is a subject in the sentence, O means object, X suggests roles other than subject or object, and – represents the absence of the entity. Thus,

| | sequel | Minister | total | episodes | variation | title | series | answer | question | character | Hacker | Radio | play | Gold |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | s | x | - | - | - | - | - | - | - | - | - | - | - | - |
| **2** | - | - | o | x | - | - | - | - | - | - | - | - | - | - |
| **3** | - | - | - | s | o | o | o | o | o | o | x | - | - | - |
| **4** | - | - | - | s | - | - | o | - | - | - | - | o | s | o |

**Figure 3.4:** The entities presence and their grammatical roles for each sentence in **??**.

| | ss | sx | so | s- | xs | xx | xo | x- | os | ox | oo | o- | -s | -x | -o | -- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0.0238095 | 0 | 0 | 0.0238095 | 0.0238095 | 0 | 0 | 0.047619 | 0 | 0 | 0.0238095 | 0.142857 | 0.0238095 | 0.047619 | 0.214286 | 0.428571 |

**Figure 3.5:** The embedding vector for text in **??**, calculating the probility for each transition according to the entities transition form3.4.

the whole column of an entity shows the information of the same entity in different sentences in discourse and each row shows the entity information of each sentence.

For the training of our later classification task, we need a numerical embedding feature to represent a discourse, for which we used the probability of each entity transition with a length no more than two in a discourse. An entity transition is a subsequence part of a column in the entity grid, which shows the presence and grammatical roles of one entity in several adjacent sentences in a discourse. The probability representation shows the distribution of entity transition in a text. And we can assume that these transitions insatiately show the local coherence of a text, and undoubtedly suggests the information about global coherence.

### 3.2.2 Selectional preference

In the area of natural language processing, selectional preference refers to the tendency of selecting particular arguments by predicates. The definition may be confusing. Let us look at an example: Have a look at the following two sentences: i) The girl is eating a cake. ii) The efficiency is eating a feeling. It is easy to tell the differences between the two sentences. The first sentence is obviously fine, but the second one does not make any sense. The efficiency is not a thing that can 'eat', while a feeling is also not a thing that can be 'eaten'. This example shows the differences between sentences with different selectional preference performance. For our text generation analysis, we want to see the performance of generated text in selectional preference, which makes a sentence make sense to human.

We employed a neural network approach for the acquisition of selectional preference by Cruys [45]. This approach gives a neural network model which computes a score of selectional preference for a set of a predicate and an argument. A connected vector of the embedding vectors for the predicate and argument is used to represent the selectional preference tuple. And the connected vector is used as the input for a feed-forward neural network to predict the selectional preference score. The structure of the network is shown in the following figure:

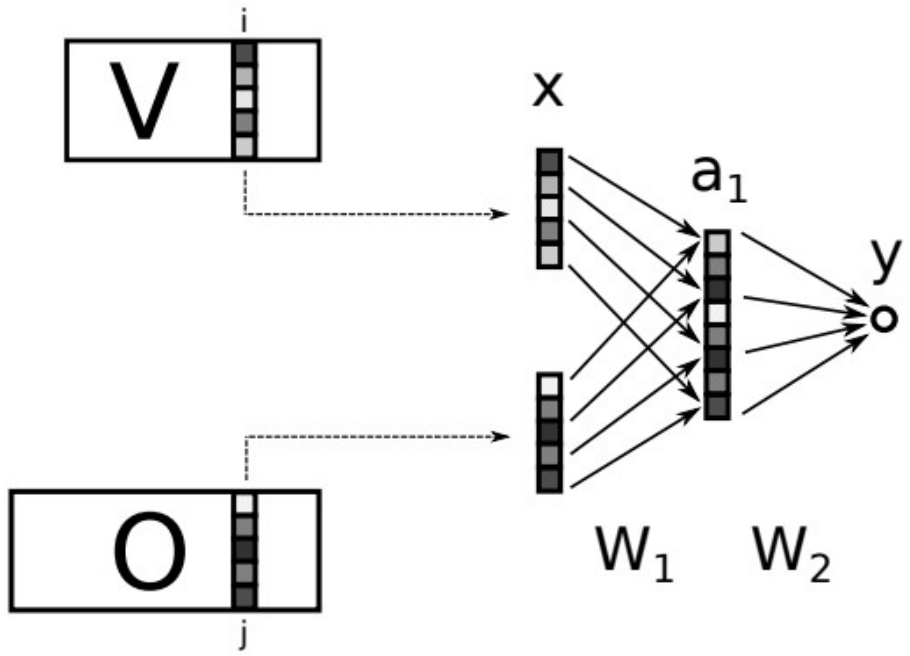



**Figure 3.6:** The structure of selectional preference ranking network

$$\sum_{j' \in J} \max(0, 1 - g[(i, j)] + g[(i, j')])$$

**Figure 3.7:** To minimize in the network structure

To make the model learn the differences between a good tuple of predicate and argument from a bad one, we construct a wrong tuple for each proper tuple by replacing the argument of the proper one with a random word. Use the tuple (i, j) to represent the proper tuple and (i, j') for a wrong one. Our goal for the model is to get a higher score g[(i, j)] for the proper tuple than g[(i, j')] for the corrupted one. Thus, the objective function for the neural network to minimize is the following:

### 3.2.3 Subject analysis by text summarization

Text summarization tasks have always been a high attention in natural language processing due to its broad application and increasing importance in the explosive period of information. Besides the practical function like saving reading time or making it easier in information retrieval, we want to employ the linguistic function of text summarization, which simplifies the text and extracts the main idea of the whole text. In the real world, human written text almost always has a clear main idea through out the whole article. And we want to analyse the ability of maintaining a subject topic in machine generated text to see if there are any differences with the human-written ones. To do the subject analysis, we introduce an unsupervised text summarization approach using sentence embedding by Padmakumar and Saran [42] and adapt it to fit our research purpose.

This unsupervised text summarization approach is based on sentence embeddings. There are various ways to encode a sentence. The simplest way is to add up the corresponding word vectors of each word in the sentence and then average them, which is similar to the way of fastText. On this basis, a little more complicated, different words have different weight. For example, the weight of a word like "and" should be lower. At this point, relevant information of tf-idf can be used as the weight [43]. However, in the above method, the order of words in a sentence is not considered, which may give a very good summarize of the model performance. Thus, the skip-thought sentence encoder [44] which takes the order information into account is adopted.

There are two parts that make up the skip-thought sentence encoder, the encoder and decoder network. The encoder is a GRU-RNN. The final hidden state of the GRU unit is passed to multi dense layers after reading the entire sentence to obtain a representation vector of a fixed length for each sentence in the input. And the coded representation become the input of the decoder network. There are two GRU-RNN

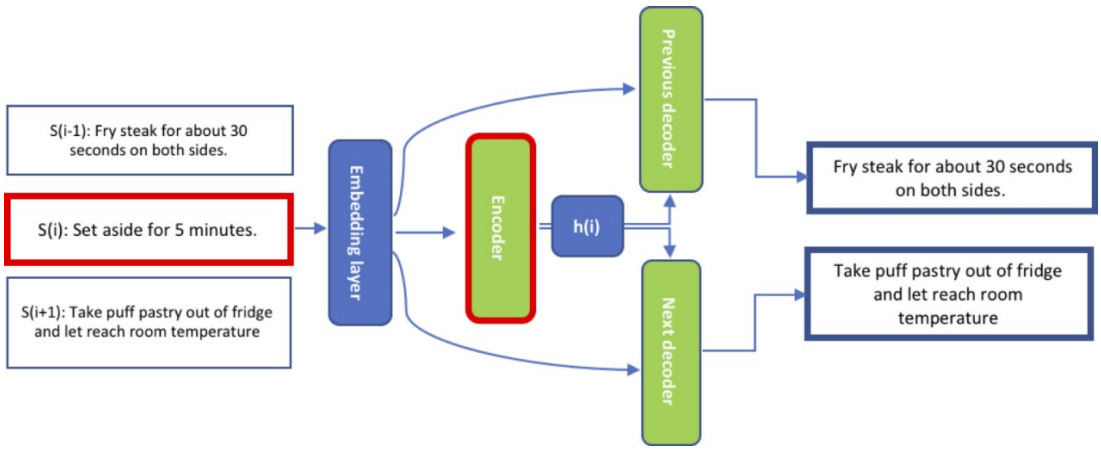



**Figure 3.8:** The working structure of encoder and decoder network in skip thoughts encoder model

decoders implemented for generating the previous and following sentence of the input sentence. And the whole encoder is working as shown in the following figure:

Given a data set containing a series of sentences, the decoder generates the previous and following sentences. The purpose of our encode network training is to minimize the sentence reconstruction loss. In this process, the encoder learns to generate vector representation, which contains sufficient information for the decoder to generate adjacent sentences. These learning representations make the embedding of semantically similar sentences closer in the vector space and therefore suitable for clustering.

In the original text summarization task, the next step is to cluster the embedded sentences in the vector space, and choose the sentences closest to the cluster center to include in the summarization text. but for our experiment, we do not really need to have the summarization text. Instead, we simply use the embedding vectors and the cluster result for the subject analysis by supposing that the text with tighter clusters has more clear main idea for the whole text.

### 3.2.4 Classification method

After the linguistic feature extraction, next comes to the real classification part for the task. For the classification method, we have tried different approached from deep learning neural network to different models in machine learning. For the neural network approach, we have used a five layers LeNet for the classification, taking the fixed size of vectors as input. We have used a dataset with 200 data, in which half is the generated text and the other half is the human written ones. We separate the dataset into two parts, one with 180 data to be the training set and the remaining 20 data to be the validation set. We also test the vectors contribution on classification using machine learning methods. In order to get a broad analysis, we implied nine different machine learning algorithm which are popular and have different advances. These algorithms includes SVM [], decision tree [], naïve bayes [], K nearest neighbors

[], bagging KNN classifier [], bagging decision tree classifier [], random forest [], adaboost [] and gradient boosting classifier []. By applying classification task on all these different models we can get a comparison for the contribution of linguistic features and the classification result with less stochastic errors.

Chapter 4

# Experimental Methodology and Results

## 4.1 Dataset description

Text generation via the neural network approach to a large extent depends on the training data. The machine learns from the data given to it, and imitates the pattern from the data. Thus, choosing the dataset used for training the GAN model is a big part of the experiment for model performance. There are many contributory factors for the experimental result, like the length of text, vocabulary size, and the subject of text. We have used five different datasets for our experiment. The following is a brief summary of the datasets.

**Image COCO**. Image COCO is a dataset given in the Texygen platform (). It is a part of the COCO dataset created by Microsoft [19], which is a dataset that includes images and the segment, caption and other features of images. The COCO dataset is widely used in the computer vision field. Here we ignored the image information of COCO dataset, simply used the description of image in the image captions to build the image COCO dataset. The description in image COCO dataset are all one sentence, which is not long and always an explanatory declarative sentence.

**News Headlines**. The dataset with the shortest text in the used datasets. The News Headlines dataset is including the news headlines from the Australian news source ABC (Australian Broadcasting Corp.) published over a period of 15 years from 2003 to 2017 [18]. Text in the dataset are all news headlines, which is short, with a length of around 10 words. That is the shortest text used in our experiment.

**Food review**. The food review dataset includes reviews of fine foods from amazon website, which covers a period over 10 years. Besides the plain text review, which is what we used in the text generation training, the dataset also contains information about products, users and the rating [15]. The food reviews are written by different consumers. Thus, they contain different language habits, and are more personal and colloquial. The length of text also varies, but generally around 150 words.

**Essay scoring**. The essay scoring dataset is the dataset used for the automated essay scoring competition [17]. It contains essays written by students in grade levels ranging from grade 7 to grade 10. The length of essays is longer than other text,

ranging from length of 150 to 550 words. Essays written by students are more standard, despite simple they may be. And an essay would be a longer text with a key message, which is valuable for us to test the contribution of main idea in classification of generated text.

**Movie summaries**. The movie summaries is a dataset of movie plot summaries and associated metadata [16]. Movie summaries are well organised long texts focused on the plot of a movie. It a little challenging for the text generation network because of its length and large vocabulary size. But its narrativity and compatibility also makes it typical for the generated text classification.

## 4.2 Generation setting

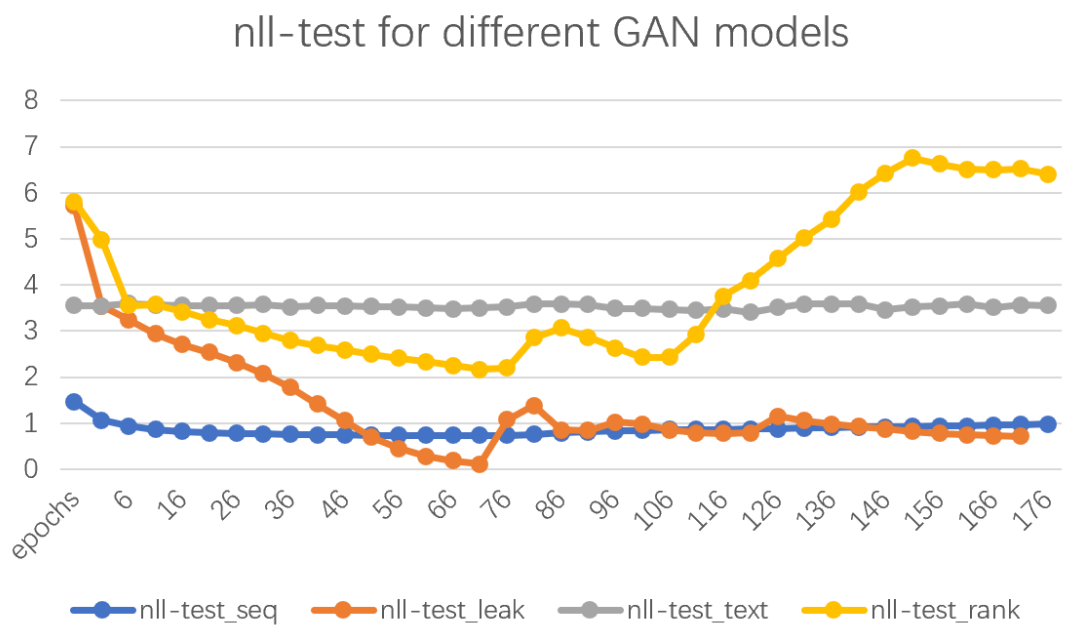
In our experiment, we want to do research on both the generation and classification part in automatic text generation process. For the generation part, we have employed the Texygen platform including different GAN models for text generation. these different models included in the Texygen platform has different advance in text generation. according to the experiment result given in the Texygen paper [3], we have used four of the given models with better performance, the SeqGAN, RankGAN, LeakGAN, and TextGAN. The dataset we used in the comparison for the performance of different GAN models is the image COCO dataset. We have selected 20000 texts in the image COCO dataset, half of which is used as training set and the other half is used for test.

For the training process, we trained 180 epochs in total. All the generators have default parameters initially follows the gaussian distribution. We started the training process with 80 epochs for the generator and 80 for the discriminator, where the MLE model is used as the pretrain process. Then after the pretrain process, it comes the 100 epochs of adversarial training. Every time we updated the generator, the discriminator is updated for 15 mini batch gradients in every epoch in the adversarial training.

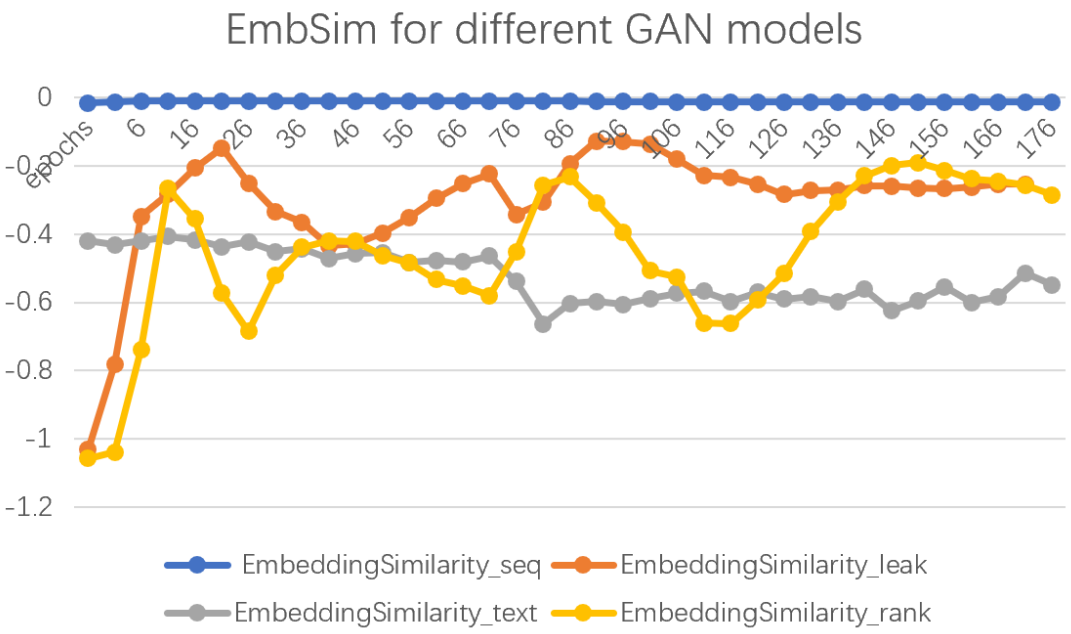
As for the evaluation matrix, we have chosen the NLL test data and Embedding Similarity to evaluate the performance of different models. The NLL is an abbreviation for "negative log likelihood". It works the same way as the maximum log likelihood. However, since most optimization problems usually try to minimize a function, negative log likelihood is preferable to maximum log likelihood. And its value reflects the performance of a text generator in generating real text. The EmbSim (Embedding Similarity) value is calculated for the word in generated text to represent the similarity of the two text, where the cosine value of the two word embedding vectors is employed in the calculation.

The figures below show the result of our training process on different GAN models with the same image COCO dataset.

In these figures, the trend of the lines shows the changing process during the training epochs. From the figures we can tell that the SeqGAN performs best on the image COCO dataset. At the latter epochs, where the model has reached a balance,




(a) nll-test scores for different GAN models




(b) EmbSim result for different GAN models

**Figure 4.1:** The generation performance of different GAN models on image COCO dataset

**Table 4.1:** Classification result test on different classification models with different embedding vectors.

| Input Structure | Vector | Vector + Coherence embedding | Coherence Embedding |
|---|---|---|---|
| **LeNet** | 0.5833 | 0.6722 | 0.5056 |
| **SVM** | 0.3 | 0.4 | 0.35 |
| **Decision Tree** | 0.9 | 0.95 | 0.85 |
| **Naive Gaussian** | 0.9 | 0.8 | 0.25 |
| **K neighbor** | 0.7 | 0.95 | 0.3 |
| **Bagging knn** | 1.0 | 0.95 | 0.5 |
| **Bagging tree** | 1.0 | 1.0 | 0.9 |
| **Random forest** | 0.9 | 1.0 | 0.45 |
| **Adaboost** | 0.95 | 1.0 | 0.8 |
| **Gradient boost** | 1.0 | 0.95 | 0.8 |

the SeqGAN gives the best performance on both NLL test score and EmbSim value. Also the LeakGAN also reached an outstanding NLL test score in the end, it takes more time and have more fluctuations in the beginning of the training, which may cause bias when not having enough training epochs. In consideration of our datasets, which are average small size with not long text, we decided to use the SeqGAN in which good performance and high efficiency can go hand in hand.

## 4.3 Linguistic feature extraction and classification setting

In the second part, we introduced linguistic features in the generated text classification task. The first feature we explored is the summarization feature of text. For the summarization feature of a text, we employed the skip thoughts model for sentence embedding, and use the cluster center of sentence vectors to represent the whole text. The degree of dispersion shows whether the text is focusing on a main point or not.

To extract a summarize for a text, we need to have a long text with a few sentences. In our datasets, we have chosen the dataset of Essays who has the proper length that meets our requirement for extracting summary information. In our generation task previous, we have got 10000 generated text for the essay dataset. We randomly take 100 texts from the generated text to get an extraction of summarization feature.

We take the skip thoughts features as baseline features for our classification task. And compared it with the embedding vector combining with the coherence embedding features to see if the coherence feature can give any positive effects on the classification tasks. The following table shows the classification result with different embedding features:

From the table we can see, among the three different representation of feature vectors, the vector combines the summarization vector and coherence vector performs

the best. The joint of coherence embedding vector makes arise the accuracy for almost every approach in the classification task. Thus, we can draw the conclusion that looking into the coherence feature of a text help us in generating a better classification model. However, the coherence feature can not be the determinant feature for a classification task. The feature model using the coherence feature only shows the worst performance in all the three feature types. It may because that the coherence feature provides one of the important features for a text but can not give a wholistic view of the whole text.

We also have made a comparison of the linguistic features in the generated texts and real human written texts. It shows that the generated text performs poorer than the real text in all the three different perspectives. The generated text is more decentralized in the summarization vector space and less proper in the selectional preference performance. These differences tell that the generated data here is not comparable with the real data.

Chapter 5

# Conclusion

According to the experiment result shown before, we can see the infect of the provided linguistic features, the coherence, summarization and selectional preference, on classification tasks. These features have made positive contributions in the experiment. The coherence embedding is a good way for text embedding to improve the performance for text classification tasks. But we can also tell that they can not be the determinant for the classification performance. Simply use the linguistic features for classification could not maintain the good performance. The linguistic features can make improvements to basic models, but can not perform the state of the art result on them own.

The linguistic features also help us to investigate the generation model and the performance of generated text. We can tell that the generated text behaves differently to the real human written text in all these three aspects. The differences indicate the gap in machine generated text and real text. The generated text can not be taken as a perfect imitation for human written text in the scope of this paper.

## 5.1 Future Work

The result of our experiment is not the state of the art result due to the lack of dataset and training time in our experiment. For example, due to the limited size of training data, we could not get a well performed model for the preference selectional evaluation, which we will improve in the future work. But the limited experiment result can still show the trend of the problem. And the resolution capability of these features also suggests a good way for generating better performed text by introducing these features into the generating process. For the future work of this project, we will extend this research to larger size of dataset, and include more different linguistic features into the scope to get both a better generation and a better classification models for text generation.

Chapter 6

# References


[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, 2014, pp. 2672–2680.

[2] Karras T, Aila T, Laine S, et al. Progressive growing of gans for improved quality, stability, and variation[J]. arXiv preprint arXiv:1710.10196, 2017.

[4] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In AAAI. 2852–2858.

[5] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial Ranking for Language Generation. arXiv preprint arXiv:1705.11001 (2017).

[6] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial Feature Matching for Text Generation. arXiv preprint arXiv:1706.03850 (2017).

[7] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Long Text Generation via Adversarial Training with Leaked Information. arXiv preprint arXiv:1709.08624.

[8] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," arXiv preprint arXiv:1701.06547, 2017.

[9] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in AAAI, 2017, pp. 2852–2858.

[10] S. Rajeswar, S. Subramanian, F. Dutil, C. J. Pal, and A. C. Courville, "Adversarial generation of natural language," CoRR, vol. abs/1705.10929, 2017. [Online]. Available: http: //arxiv.org/abs/1705.10929

[11] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," CoRR, vol. abs/1709.08624, 2017. [Online]. Available: http://arxiv.org/abs/1709.08624

[12] Z. Yang, W. Chen, F. Wang, and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets," arXiv preprint arXiv:1703.04887, 2017.

[13] Z. Yang, J. Hu, R. Salakhutdinov, and W. W. Cohen, "Semisupervised qa with generative domain-adaptive nets," arXiv preprint arXiv:1702.02206, 2017.

[14] A Huang H, Yu P S, Wang C. An introduction to image synthesis with generative adversarial nets[J]. arXiv preprint arXiv:1803.04469, 2018.

[15] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013. http://i.stanford.edu/ julian/pdfs/www13.pdf

[16] Learning Latent Personas of Film Characters. David Bamman, Brendan O'Connor, and Noah A. Smith. ACL 2013, Sofia, Bulgaria, August 2013

[17] https://www.kaggle.com/c/asap-aes/data

[18] Rohit Kulkarni (2017), A Million News Headlines [CSV Data file], doi:10.7910/DVN/SYBGZL, Retrieved from: [this url] https://www.kaggle.com/therohk/million-headlines

[19] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//European conference on computer vision. Springer, Cham, 2014: 740-755.

[20] Reiter E, Dale R, Feng Z. Building natural language generation systems[M]. Cambridge: Cambridge university press, 2000.

[22] https://www.jiqizhixin.com/articles/2017-05-22

[23] Reiter E. An architecture for data-to-text systems[C]//Proceedings of the Eleventh European Workshop on Natural Language Generation. Association for Computational Linguistics, 2007: 97-104.

[24] Mei H, Bansal M, Walter M R. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. arXiv preprint arXiv:1509.00838, 2015.

[25] Huang THK, Ferraro F, Mostafazadeh N, et al. Visual storytelling[C]// NAACL HLT. 2016.

[26] Rush A M, Chopra S, Weston J. A neural attention model for abstractive sentence summarization[J]. arXiv preprint arXiv:1509.00685, 2015.

[27] Chopra S, Auli M, Rush A M. Abstractive sentence summarization with attentive recurrent neural networks[C]//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 93-98.

[28] Rush A M, Chopra S, Weston J. A neural attention model for abstractive sentence summarization[J]. arXiv preprint arXiv:1509.00685, 2015.

[29] Gupta A, Agarwal A, Singh P, et al. A deep generative framework for paraphrase generation[C]//Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[30] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

[31] Wang B. Disconnected Recurrent Neural Networks for Text Categorization[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 2311-2320.

[32] Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification[J]. arXiv preprint arXiv:1607.01759, 2016.

[33] Lei Z, Yang Y, Yang M, et al. A multi-sentiment-resource enhanced attention network for sentiment classification[J]. arXiv preprint arXiv:1807.04990, 2018.

[34] Zhang Y, Gan Z, Carin L. Generating text via adversarial training[C]//NIPS workshop on Adversarial Training. 2016, 21.

[35] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[36] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences[J]. arXiv preprint arXiv:1404.2188, 2014.

[37] Lai S, Xu L, Liu K, et al. Recurrent convolutional neural networks for text classification[C]//Twenty-ninth AAAI conference on artificial intelligence. 2015.

[38] Yang Z, Yang D, Dyer C, et al. Hierarchical attention networks for document classification[C]//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 1480-1489.

[39] Reiter, Ehud and Robert Dale. 2000. Building Natural-Language Generation Systems. Cambridge University Press, Cambridge, England.

[40] Asher, Nicholas and Alex Lascarides. 2003. Logics of Conversation. Cambridge University Press, Cambridge, England.

[41] Barzilay R, Lapata M. Modeling local coherence: An entity-based approach[J]. Computational Linguistics, 2008, 34(1): 1-34.

[42] Padmakumar A, Saran A. Unsupervised Text Summarization Using Sentence Embeddings[J]. 2016.

[43] Arora S, Liang Y, Ma T. A simple but tough-to-beat baseline for sentence embeddings[J]. 2016.

[44] Kiros R, Zhu Y, Salakhutdinov R R, et al. Skip-thought vectors[C]//Advances in neural information processing systems. 2015: 3294-3302.

[45] A Neural Network Approach to Selectional Preference Acquisition

[46] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[47] Girju R. Automatic detection of causal relations for question answering[C]//Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12. Association for Computational Linguistics, 2003: 76-83.

[48] Barzilay R, Elhadad M. Using lexical chains for text summarization[J]. Advances in automatic text summarization, 1999: 111-121.