ROBERT GORDON UNIVERSITY ABERDEEN

SCHOOL OF COMPUTING SCIENCE AND DIGITAL MEDIA

Honours Project

Student Name: Craig Davies	Matriculation Number: 0702555
Supervisor: Prof. Nirmalie Wiratunga	Second Marker: Dr. Stewart Massie
Course: BSc (Hons) Computing: Application Software Development	
Project Title: A Quality Metric for Neural Lyric Generation	
Start Date: 01/10/2017	Submission Date: 03/05/2018

CONSENT

I agree ■
I do not agree ☐

That the University shall be entitled to use any results, materials or other outcomes arising from my project work for the purposes of non-commercial teaching and research, including collaboration.

DECLARATION

I confirm:

- **That the work contained in this document has been composed solely by myself and that I have not made use of any unauthorised assistance.**
- **That the work has not been accepted in any previous application for a degree.**
- **All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.**

Student Signature: Craig Davies	Date Signed: 03/05/2018

RGU SYMBOL OF SUCCESS

A Quality Metric for a Neural Lyric Generator

Craig Robert Davies

A report submitted as part of the requirements for the degree of BSc(Hons) in Computing: Application Software Development at the School of Computing
Robert Gordon University
Aberdeen, Scotland

May 2018

Supervisor Prof. Nirmalie Wiratunga

Abstract

This project reviews whether song lyrics written by Artificial Intelligences(AIs) can pass a Turing-like Test. AIs have been used in the past to write poetry that ostensibly passed the Turing test, but poetry tends to be less comprehensible to the layman than song lyrics, and thus song lyrics provide a more rigorous trial of modern AI's creative capabilities. A new metric was created taking into account the percentage of words written in natural English and the number of grammatical errors to rate the quality of AI-generated song lyrics. The new metric was calibrated with human responses. A Recurrent Neural Network (RNN) was created with different training sets and network parameters. From each of thirty-two trained states, the RNN wrote the lyrics to 5 songs, each of which was 200 words long, of those five, the highest rated by the new metric was kept. Eight sets of song lyrics of varying quality were selected to show to users. Users were shown each set of lyrics one at a time interspaced with human-written avant-garde lyrics and then asked to guess whether lyrics shown were written by a human or a bot, bot being used as a colloquial term for AI. Passing the Turing test is defined as being mistaken for human 30% or more of the time under specific conditions. Four of the bot-written lyrics passed for human-written lyrics over 30% of the time including the three highest rated songs. These four songs were then shown one at a time alongside one of the avant-garde lyrics from the previous test to different users. These users were asked to guess which of the two sets of shown lyrics were written by a bot. Three of the four bot-written lyrics passed for human over 30% of the time including the two highest rated songs written by the RNNs.

Acknowledgements

I would like to thank everyone who I met and spent time with at University, I could not have gotten through it without the encouragement and support of my peers.

In particular I would like to thank Professor Nirmalie Wiratunga for her input and suggestions along the way and for being the one to realise what should be the true focus of my honours project; Dr John Isaacs for really having the backs of everyone who was working through what turned out to be quite a challenging year; Dr Niccolo Capanni for helping me realise I have the potential to do what I want to do; my mother Lesley Davies for her strength and support and my girlfriend Lucia Suhanyiova for her patience, understanding and encouragement.

Declaration

I confirm that the work contained in this BSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Date: 3/5/18 Craig Robert Davies

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

The Turing Test is a measure of whether an Artificial Intelligence (AI) can accurately imitate a human being (Oppy, 2003). As modern AI becomes more powerful, it is becoming more ubiquitous than ever before. It is also far more likely that AI will pass the Turing Test in more ways than which it already has (Rapaport, 2003). At its inception, it was unclear how successful this project would prove to be. The initial plan was a rough template-based song lyric generator creating lyrics based on commonly used words in songwriting, combined with a grammar correction algorithm that would hopefully result in something resembling human-written song lyrics. However, the literature revealed that there were more efficient solutions (Vinyals et al., 2015) and that if a machine could write song lyrics that would pass the for human-written lyrics, they would have likely been created through a methodology involving a neural network. Whenever the word bot is used, bot is referring to a neural network. Whenever the word song is used, song is referring to the lyrics to a song.

1.1 Motivation

Even if modern neural networks are still not capable of passing the Turing Test in this way, song lyric generation is an accessible way to introduce people to the power of neural networks as they are something that everyone can recognise and understand without introduction or explanation. Ultimately then, this project was not just a test of my abilities, it was an exploration of where modern AI lies on the road to human-level performance and beyond.

1.2 Legal, Social, Ethical, Security and Professional Issues

1.2.1 Legal Issues

No personal data of any variety was stored so there are no legal issues to mention, if this system was used commercially, artists might take issue with some of the similarities between songs produced by the RNN and the original dataset. Some of the more repetitive Aerosmith songs, in particular, had a substantial influence on the songs generated by the RNN.

1.2.2 Social Issues

If this project proved monumentally successful, it could have a disruptive effect on the creative industries. More likely the tools generated in this project would be used to aid rather than hinder those working in the creative industries.

1.2.3 Ethical Issues

As above, if this project was successful enough that it was able to put people out of work, this could be considered ethically questionable. However, this would be a misuse of the project rather than an issue with the project itself.

1.2.4 Security Issues

There are no significant security issues with this project, no sensitive data of any kind was used in this project.

1.2.5 Professional Issues

Nothing about this project could bring the profession into disrepute. However, care was made to reference all sources of data or software that were used during the creation of this project.

1.3 About this Thesis

This is the thesis of *Craig Robert Davies*, submitted as part of the requirements for the degree of BSc (Hons) Computing: Application Software Development at the School of Computing, Robert Gordon University, Scotland.

The purpose of this thesis is to propose a new simple automated metric for machine-written song lyric evaluation and to create song lyrics in several ways with which to test and refine the new metric.

1.4 Chapter List

Chapter 2 Background Research: A broad review of the literature regarding Natural Language Generation and Automated Metrics.

- Recurrent Neural Networks and Natural Language Generation
- Creative Natural Language Generation
- Template-Based Creative Natural Language Generation
- Recurrent Neural Network-Based Creative Natural Language Generation
- Other Techniques for Creative Natural Language Generation
- Existing Evaluation Metric Quality

Chapter 3 Requirements Specification: The broad functional and non-functional requirements for all the aspects of this project.

Chapter 4 Design: How the specific elements of this project were initially designed.

Chapter 5 Development Practices: Design Philosophies for the entire project.

Chapter 6 Implementation: Exactly what was produced and how it was made.

Chapter 7 Results: The final results of every test.

Chapter 8 Limitations and Evaluation: Limitations of the project and an evaluation of project quality

Chapter 9 Conclusion: The conclusions of the thesis are presented.

Chapter 2

Background Research

2.1 Introduction

This Chapter reviews how natural language is used with regards to creative writing and what role Recurrent Neural Networks (RNNs) have in the future of Natural Language Generation (NLG). Natural language processing is a promising sub-field of Artificial Intelligence (AI), the end goal of which is to create text that can be read as if it was written by a human (Oliveira, 2009). Early NLG has been attempted for decades (Bourbeau et al., 1990), but we are only really approaching human-level performance in the past few years (Vinyals et al., 2015). Some algorithms and methods have beaten human performance by specific metrics, and it is probable that this is just the beginning (Oliveira, 2009). Natural Language Generation may be one of the last fields in which human beings can still outperform Artificial Neural Networks, but it appears that our dominance in this area will come to an end soon enough (Oliveira, 2009). Recently Recurrent Neural Networks and Convolutional Neural Networks (CNNs) have begun to prove dominant in the field of NLG, and their rapid recent improvements show no sign of slowing (Vinyals et al. , 2015). However, AI's performance in the realm of NLG is primarily restricted to more straightforward, shorter tasks, like object recognition and caption generation. When put to more substantial or more complex functions like poetry generation, AIs still perform significantly worse than the best human beings and progress here is slow (Oliveira, 2007,2015). Significant advantages that AIs have over human beings is their ability to produce significant quantities of work very quickly, and it is plausible that in the future people will use AIs to enhance their work by getting an AI to generate many ideas that the person will then look through and choose one to use.

2.2 Recurrent Neural Networks and Natural Language Generation

RNNs are increasingly becoming the dominant form of producing high-quality NLG. Although they have not frequently been used for creating writing yet, they have been used for more straightforward NLG tasks like caption generation.

Probably the best-known example of modern successful RNN-based NLG is Google's Show and Tell image caption generation system (Vinyals et al. , 2015). Show and Tell attempted to automate image caption generation. Image caption generation is a particularly tricky skill for AIs as it involves several skills that historically struggled to perform well. Analysing unseen images and NLG have both been challenging tasks for previous AIs to perform (Vinyals et al., 2015). The idea that this task could be automated is appealing and not just for accessibility purposes. In today's image-centric society, millions of images are uploaded to the image sharing platform Instagram, every single day (Etherington, 2013). Instagram is only one of many image sharing platforms too, the total number of new pictures uploaded to the internet each day is practically inconceivable. If we wanted every one of these images to be accurately captioned, the human resources required would be unfathomable. An AI that could automate this task would save countless staff-hours. The model works by using a pre-existing state of the art CNN to view and analyse the image. The data from the analysis was then sent to the RNN for caption generation. The model was trained with several training sets of images. After each set, its performance was measured by a combination of the BLEU-1, BLEU-4 (Papineni et al.,2002), METEOR (Banerjee and Lavie., 2005) and CIDER (Vedantam et al., 2015) metrics. The performance of the model was marginally better than human-level performance when tested by the CIDER metric after training with the MSCOCO (Lin et al., 2014) development set. Even more impressive, the model performed 27% better than human-level performance when measured by the BLEU-4 metric on the same dataset. The model did not quite reach human-level performance by any other metric, or after training with any of the other data-sets, nevertheless, the fact that it could achieve better than human-level performance by any metric at all is impressive. However, automated Metrics are not as useful for measuring a neural network's performance as human-based metrics (Novikova et al., 2017). In general, the larger the set that the RNN was trained with, the better the model performed compared to human-level performance, so it is likely that with the superior training sets that the future will provide, this model will surpass human-level performance consistently and entirely.

This problem domain was analysed one step further (Venugopalan et al. 2014), and in 2014 attempts were made to generate captions for videos rather than images. A video is just a series of slowly changing images, so the overall problem is similar. Very few attempts have been made at generating image captions for videos "in the wild" i.e. videos where it was not known in advance what objects are going to feature in the video or what actions are going to take place. It is also exceedingly difficult for a neural network to comprehend how a series of images are related to each other. As such, they use the surrogate description of Subject (who or what is performing an action) verb (what they are doing) and object (what the subject is doing something with) collectively called SVO. Similar to the Show and Tell scenario, every frame in the video was analysed by a CNN and then fed to a deep RNN for translation into Natural Language. Doing this with every frame seems computationally excessive, using every other frame would be unlikely to result in significantly lower quality and would of course double the speed of the model, but this requires further research. The CNN was pre-trained with a batch of 1.2 million images each with category labels to determine what is in them; the RNN was trained with 100k captioned images. They tested their model's BLEU and METEOR scores as well as evaluating what percentage of the time their model correctly guessed the SVO of the videos; these metrics were compared against what were the state of the art models at the time. The resulting video captions were then put to human-testing to evaluate their relevance to the video at hand as well as their grammatical correctness; these captions were also compared against the captions written by the previously state of the art model as well as human-written captions. They compared their model with different training sets and predictably, the more the model was trained and the larger the dataset the model was trained with; the better the model performed. This matches the findings from other papers (Vinyals et al., 2015). The model beat the previous state of the art in BLEU, METEOR, Verb, Object and relevance metrics, losing out slightly in the grammar and Subject metrics.

2.3 Creative Natural Language Generation

Since people started experimenting with neural networks, they have tried to use them to create natural language (Bourbeau et al.,1990). The potential social utility of getting this right is enormous. Whether we use neural networks on their own to develop NLG, like in image caption generation (google, 2015) or whether we use them as a tool to enhance our abilities for example by assisting with simpler IT helpdesk queries with an AI as the front line. Alternatively, using an AI to generate sample lines of poetry or song lyrics and then using that as a skeleton on which to write original text. The question that remains to be answered is whether an AI can ever really be creative or whether it merely follows rules that it has either been taught

or has learned for itself (Dartnall, 2013). This question can be expanded further by asking what it means to be creative. However, any criticism that can be made of an AIs creativity can equally be used to criticise the creativity of the human mind. Early attempts at creative natural language generation focussed on rules and templates, this allowed the creation of some unique, but ultimately formulaic original text (Oliveira, 2009).

Neural Networks have been used to create wholly original poetry and song lyrics, but different approaches have had results that differed significantly in quality. Most attempts at using an AI for creative writing have been made with template-based or rule-based approaches (Oliveira, 2009), however recently progress has been made, and some success has been achieved using RNN based approaches. It is probable that this improvement will continue and that RNN based methods will be state of the art in this field within a relatively short period (Xie, 2017).

2.4 Template-Based Creative Natural Language Generation

Template-based approaches to NLG have been attempted for decades now (Bourbeau et al., 1990) but most of these attempts had relatively little success, at least regarding passing the Turing Test or generating original text of high objective quality. Bourbeau et al. merely used real poetry and replaced words with other syntactically correct words to create syntactically correct though not semantically correct, ostensibly original poetry.

Gonalo Oliveira has made several attempts at this over the years with undeniable success but little measurable improvement. Oliveira's (2007) initial research, Tra-la-Lyrics (Oliveira, 2007) was a straightforward template-based system for generating original song lyrics created by using existing music and breaking the beat down into patterns (strong beats, rests). Words were then assigned to the pattern with strong syllables occurring on strong beats and weak syllables occurring on weak beats. Three different strategies were used for the generation of the lyrics. The Random Words strategy just generated any random word that fit the rhythm. The generative grammar strategy wrote words that best fit pre-written grammar templates, with each category of word(noun, verb) occurring at the correct point in the template. The Generate and test strategy is more complicated than generative grammar. Sentences were generated one at a time according to the grammar templates mentioned above.

Generated sentences are then scored by how well they fit the rhythm, whether they rhyme with other lines and whether they contain the correct number of syllables, a new line generated with a higher score replaces whichever line has the lowest score. This process is then iterated until a set number of generations has been completed, or a maximum theoretical feasible score has been reached, this strategy takes considerably longer than the others. The lyrics made grammatical sense, but there was very little that was done to ensure that they made sense semantically. Thus the songs that were created worked rhythmically and grammatically but did not have a coherent theme. The songs generated by Tra-La-Lyrics were evaluated by human beings based on their rhythm, rhyme, semantics, sound and overall quality, with each strategy being assessed separately. This data is useful because as well as showing a total quality score for each algorithm, it is also clear which perform best and worst in each area, thus allowing improvements to focus on the areas of greatest weakness. Overall the generative grammar model performed the best, this is curious and could be due to an issue with the rating system, it is likely that the weights of the variables were arbitrarily chosen and were not calibrated with human users. The generate and test strategy may have also underperformed because it overwrote whichever line had the lowest score, although this line was low-scoring, it could have still contributed to the overall rhyme or rhythm of the song. The conclusion that the generative grammar model was the best thus far led to further work on that model, though it is probable that the generate and test strategy would have performed better with refinement.

Oliveiras next attempt at poetry with PoeTryMe (Oliveira, 2012) worked similarly to the attempt at writing song lyrics. Poetry is given more artistic licence than other forms of NLG in general, and thus minor grammatical failures or semantical obscurities can be passed for creative licence. This makes writing poetry an ideal task to test the capabilities of modern NLG systems. The poetry generator that was created was broken down into a sentence generator, a template generator and a grammar generator. They were broken down to make the tools as modular as possible and as simple to alter as possible for future use cases. In theory, this system can be adapted for use with any creative NLG. Modular components can also theoretically be tuned as needed on the fly to work out how changes to the different components can be used to improve the overall product or create different types of end-product. Potential words were generated in "triples" with the triple structure being word 1, relationship, word 2. With the triples being stored in a graph database, with the words as nodes and the relationships as the links between the nodes, this allowed the network to work through the database looking for a word that fit the combination of structure, grammar and meaning the best. With this paradigm, semantics took a back seat to grammar and grammar took a back seat to structure, whether this was

the correct approach remains to be seen. Generative Grammar and Generate, and test strategies were both utilised, and sample poetry was made available. However, the work was not evaluated in any meaningful way either by human or automatic metrics. This paper was written mostly to introduce anyone who wished to continue this work to the modular PoeTryMe model.

In 2015 Oliveira returned to this concept a third time with Tra-la-lyrics 2.0 (Oliveira, 2015). Tra-la-lyrics 2.0 was an alteration of the PoeTryMe method to take into account rhythm like the original Tra-la-lyrics, this version of Tra-la-lyrics also used seed words or phrases to generate the lyrics, with the intention that keeping the words to a theme might mean they would make more semantic sense. Three strategies were tested, the generative grammar model from PoeTryMe, a simple Rhythm and Rhyme (R & R) model that aimed only to have rhymes at the end of specific lines and have the syllables match a predetermined rhythm and Tra-la-lyrics 2.0. These models were put to human evaluation and evaluated based on their rhythm, rhyme, sound, grammar, meaning and their overall quality. Users were also asked to guess what the seed word was that allowed the generation of the lyrics. The Tra-la-lyrics 2.0 algorithm performed the best out of the three that were tested by every metric that they were tested except one, scoring the best of the three in rhyme, rhythm, meaning, sound and grammar and scoring second in whether the users could guess the theme. These results were obtained from an average of 90 user surveys for each category of song. The testers could not guess the seed word that was used as frequently as they could with the R R algorithm. The scores ranged from 2.7 to 3 out of 5 on average for each metric, and although this could potentially be compared to some song lyrics written by human beings, it is probably not comparable with the best.

2.5 Recurrent Neural Network-Based Creative Natural Language Generation

RNN-based tools for natural language generation have been used less frequently in the past because until recently they were significantly harder to produce, train and test. However, with recent developments on tools such as TensorFlow (Abadi et al., 2016) and how effective RNNs have been at other approaches (Vinyals et al., 2015), it is likely that we will see many more attempts at creative writing from RNNs in the future.

Of course, there is nothing to stop someone using a combination of approaches to generate original natural language. Ghazvininejad et al. (Ghazvininejad et al., 2016)

used a template and an RNN in unison to create original topical poetry in iambic pentameter. The model had an exceedingly large vocabulary of over 15,000 words from which to write poetry. The words were ordered by their stressed syllables with a 0 representing low stress and a 1 representing high stress, the syllables alternated between low and high stress as much as possible to create iambic pentameter. The model was given a seed word and chose a series of words that related to it. This was done by analysing a large bank of sentences and seeing how frequently various words occurred together. The formula used is loosely based on Bayes Theorem (Bayes,1763) with the frequency of a word being found near the topic or seed word being divided by the frequency of said word being found in the main text being used to calculate the relatedness of the word to the topic or seed word. The model then searched this series of related words for rhyming pairs and set them according to the pattern at the ends of alternating lines. The RNN was then fed these seed words and rhyming pairs and created a grammatically correct sentence containing several of the words related to the seed word in iambic pentameter and with rhymes on every other line. With the rhyming pairs in place, the model uses the RNN to generate a path consisting of every possible combination of words that fit the iambic pentameter constraints and contain the rhyming words. These paths are then run through by the RNN and scored. The RNN scores paths by comparing them to a large bank of song lyrics on which it was trained. The RNN was trained with over 90,000 sets of song lyrics. The results were impressive but were not objectively tested by any metrics except against other generation strategies within the same model. It would be interesting to see how people would rate the poetry generated by this model; it appears to be of very high quality. This hybrid RNN and template approach warrants further study.

Wen (Wen et al.,2015) used an RNN to generate natural language which was then evaluated and ranked by a CNN. Their goal was to create a wide-use flexible NLG system that does not depend on pre-defined grammar rules or contexts. The intended purpose of this model is primarily focussed on customer-service scenarios, with the model being able to handle simple end-user queries. The end-user would ask this model a question, and the model would use the contents of that question and previous training scenarios to generate an appropriate response. If this can be done accurately and efficiently, this is indeed appealing, having an automated, accurate response to a significant number of end-user queries would free up countless staff-hours for more complex, less tedious tasks. The example they tested it on was a restaurant finder. Restaurants and queries were given attributes (e.g. the type of food they serve/are looking for). The attributes of the user queries were then detected, the queries were performed, and then an appropriate response was generated by the model by going through both the query and the response and checking that each attribute that was

present in the query was also present and addressed in the response. The model performed better than any others that it was tested against and averaged a user rated informativeness score and a naturalness score of 4 out of 5, this is impressive and demonstrates that this model shows plenty of promise for the future. It is entirely possible that 4 out 5 would represent human-level performance, but unfortunately, the model was not compared to human responses.

2.6 Other Techniques for Creative Natural Language Generation

Template-based and Neural Network based are the primary techniques for generating natural language, but there have been attempts made at other techniques. Kumagai used a Monte-Carlo tree-search to generate natural language in 2016 (Kumagai et al., 2016). Even though nearly all the work on natural language generation in the past few years has been done using Neural Network based approaches, it shows that research is still being done on other approaches. The algorithm works by building a tree of possible words to make a sentence and then a tree of those possible trees to find the best sentence to fit a specific scenario. Sentences are generated by looking at what words probabilistically follow, or are spoken alongside other words. The sentences created were scored by their syntactic probability, their acceptability and their perplexity with higher scores on each metric being rated as better. The resulting generated sentences had little degree of grammatical correctness and very rarely made much syntactical sense, it is hard to determine from this paper if this entire approach is not viable or whether it is because they made a mistake when creating their scoring parameters. It is also possible that they were merely too lax with their scoring parameters, there seems to be no limit to the potential Acceptability or Perplexity scores, and the higher those scores are the closer the sentences seem to be to natural language. Once again the results were not put to human evaluation, so it is hard to judge objectively just how good the results were. This paper did not put their findings to standard automated metrics and instead just created their own that they use to generate and evaluate their generated sentences. This makes the results especially hard to assess objectively. With further study, tweaks to their algorithm and sentence scoring, this approach may provide results in the future, but it appears that for now, RNN-based methods perform better.

Thomason et al. (Thomason et al.,2014) used a pre-existing visual recognition technique (Guadarrama et al.,2013). They focussed on implementing a new statistical method for generating the captions for youtube videos "in the wild". The extract

two frames per second from the video, which is significantly more efficient than the method described by Venugopalan and in fact, this was the method that Venugopalan compared their approach to, with only limited improvement. This model analyses the outputs from the visual recognition data and looks at the Subject, Verb or Object that had the highest certainty. It then looks at the others and probable relations between them, if whatever has been detected seems unlikely to be related to whatever they are most certain about, it goes down the list of possibilities and chooses one that is more likely. Predictably this model performs best in situations that are likely to occur and in fact performs worse than other models in more unusual cases. The example they use is a video of a man playing the guitar on a tree, the model initially interprets this correctly and then retrofits it to be a man performing guitar on a stage as that would be far more likely. This model worked better than the then state of the art in the majority of cases and even beat the RNN-based approach in one or two scenarios, on paper it also appears to be a more efficient algorithm, but whether that stands in the real world or not remains to be seen. It goes to show that although Neural Network based approaches seem to be the best performing these days, they are not always the best fit for every circumstance.

2.7 Existing Evaluation Metric Quality

Reviewing the literature relating to NLG has shown that reliance on such a vast array of different automated metrics has meant that it was hard to find objective or meaningful measures to determine which tools were the best at which tasks. Novikova (Novikova et al.,2017) came to the same conclusion. They evaluated a large number of automated metrics as well as how they compare with human ratings. Ultimately what we are interested in is what a human being thinks of the output of the model, and this, unfortunately, does not correlate well with how well a model performs by automated metrics (Novikova et al., 2017). At the end of the day to determine whether a model is performing as well as expected, people need to be used to rate the performance of the said model. On the other hand, automated metrics are accurate at the lower end of the scale, i.e. if a model is performing poorly, automated metrics will point this out accurately. It is at the upper and more critical end of the scale where automated metrics are of limited use. It is essential to consider this when making a new model and be sure to put it to human evaluation if it is intended for the results to be seen by humans. Reliance on surrogate outcomes, like automated metrics will make it hard to distinguish the best NLG from merely passable NLG.

2.8 Conclusions

The most effective methods for creative natural language generation appear to be RNN-based methods. Most of the best-performing models for NLG systems at least consisted partly of a Recurrent Neural Network, so this is the area in which this project will focus primarily. Pattern-Based models also seem to have had some success and are relatively simple to implement from scratch. For these reasons the main tools used for the generation of song lyrics in this project will be a Recurrent Neural Network-based song lyric generator and a Pattern-based Song lyric generator. The current generation of automated metrics aren't performing as well as human-based metrics, and thus an attempt will be made to create a new automated metric that is calibrated based on human opinion, with the ultimate focus of this metric to be that it reflects the average of human opinion as closely as possible. This metric will also be designed in such a way that emphasises high usability so that anyone who so wishes will be able to implement this new metric to rate their own creative NLG as quickly and as accurately as possible. A website will be created where tests will be hosted, to determine the how frequently bot-written songs can pass for human-written songs. A Google Forms Survey will be created to rate bot-written songs and calibrate the new metric for future ratings.

Chapter 3

Requirements Specification

3.1 Broad requirements

This application is targeted at songwriters who will be able to use this paradigm to enhance their creative capabilities. Using the best approach to original-lyric generation, they will be able to find creative and original lyrics on which to base titles, verses or entire songs. This will decrease the total time that it takes to write original songs.

This project is also designed to examine which of the currently existing machine-assisted methods of original lyric generation works the best for this purpose.

The intention is also to create a new metric for measuring NLG that can be used to rate machine-written song lyrics in a way that is as simple and as objective as possible. This metric will be produced in a way that will match human opinion as accurately as possible.

3.2 Lyric Generation Modules

3.2.1 The Random Lyric Generator

- The Random Word Generator will produce a selection of random words derived from a dataset of the most commonly used song lyrics
- The Random Lyric Generator will produce a new song worth of lyrics in less than 10 seconds

The Random Word Generator will produce a selection of random words derived from a dataset of the most commonly used song lyrics.

3.2.2 The Template-Based Lyric Generator

- Templates will be generated based on popular songs
- The Template-Based Generator will produce song lyrics according to the template it has been given choosing words from the list words used in popular songs, with a strong bias towards words used most frequently occurring in the dataset
- The Template-Based Generator will generate a new song within 30 seconds.
- Songs produced by the Template-Based Generator will always be different

3.2.3 The Recurrent Neural Network-Based Lyric Generator

- Songs produced by the Recurrent Neural Network-Based Lyric Generator will be generated in 98+% English
- Songs produced by Recurrent Neural Network-Based-Based Lyric Generator will have the minimum possible grammatical errors
- Songs produced by Recurrent Neural Network-Based-Based Lyric Generator will contain some wholly original lines
- Once the network has been trained, songs will be produced within 1 minute
- Songs produced by the Recurrent Neural Network-Based-Based Generator will always be different
- Songs produced by the Recurrent Neural Network-Based Lyric Generator will resemble human-written songs as much as possible

3.2.4 The Website

- The website will feature two tests for users to take
- The first test will show the user a song and ask if it was written by a bot or human
- The second test will display two songs to the user and ask if the user can tell which one was written by a bot

- The website will load within 5 seconds under normal circumstances
- The website will save the results of the test to a database for later analysis

3.2.5 The Database

- A MySQL Database will store the results of the website tests for future analysis
- The will be able to be queried to determine if any bot-written songs have passed for human-written over 30% of the time

3.2.6 The Database

- A MySQL Database will store the results of the website tests for future analysis
- The will be able to be queried to determine if any bot-written songs have passed for human-written over 30% of the time

3.2.7 The User Survey

- The User Survey will be created using Google Forms
- The User Survey will ask the users to rate the songs on their grammar, semantics and the quality of their English as well as how likely it seems thatthe songs featured in the survey were written by a human
- The User Survey will attempt to show the value of the new metric

3.2.8 The New Metric

- The scores generated by the New Metric will represent human opinion as closely as possible
- The scores generated by the New Metric will also accurately represent how likely it is that a set of bot-written song lyrics will be mistaken for human-written song lyrics
- The User Survey will attempt to show the value of the new metric

Chapter 4

Design

What needed to be tested was whether song lyrics written by an automated system could pass for song lyrics written by a human being 30% of the time. Although existing NLG systems for creating original song lyrics already existed, the methods and datasets that they used are all vastly different. As well as rating existing systems this project set out to create a new Natural Language Generating RNN, based on an existing RNN that has been used to write plays in the style of Shakespeare (Kofler, 2017). Two other simple methods of creating original song lyrics were also built in Tandem, with the view that the RNN-based approach would prove to be superior. A template-based song lyric generation system was designed as well as a simple random word generator; both of these systems used the same dataset of song lyrics as the RNN. A method had to be devised with which to test whether the lyrics generated by this RNN and others were human enough to pass the Turing Test. Two separate tests were proposed, one where songs would be shown to the user and they had to guess whether the songs were written by human beings or an automated method. A second test was created where two songs would be showed to the user, and the user had to guess which song was written by a human and which was written by an automated system. To give an idea of how the songs were performing in quality before they were put to human ratings, a new simple automated metric was created to quickly rate songs as accurately as possible, this metric was evaluated and calibrated using the results from the Google Forms Survey and the Turing-like Tests.

4.1 The Random Word Generator

The random word generator will be a simple algorithm written in python, it will be used to generate a list of roughly 200 words that are extracted at random from a list of the words most commonly used in song lyrics.

4.2 The Template-Based Song Lyric Generator

The template-based model will be written in Python and will create an original song when provided with the template of that song. The template of the song will be a series of character representing the type of word (noun, verb) and how many syllables the word contains for example "1n" representing a one syllable noun. The model will then randomly select a word from the words of that category that most commonly occurred in the dataset that was used to train the RNN. If the word does not fit into any of the categories that the model has been provided with, the word will be provided to the template as is. For example, the sentence "the cat is brown" would be replaced with "the 1n is 1v" when provided to the template-based model. The Template-based model will write words in the same structure as they were inputted. This will mean the same numbers of the same types of words in the same order per line.

4.3 The Recurrent Neural Network-Based Song Lyric Generator

The Recurrent Neural Network-Based system will be based on a pre-existing state of the art NLG system. It will then be customised to the specific task of generating song lyrics and trained on the highest quality and most extensive possible datasets of song lyrics that are available. The dataset will be as large as is possible given the economic and time constraints. Training a neural network can take a significant length of time and only so many iterations of training and evaluating the network before the best performing model thus far will need to be selected. The model's state will be saved after each complete session of training, with the model being trained from scratch every time so that the performance of the model in different states and under different conditions can be compared after the fact. The model will be trained on portions of a dataset downloaded from Kaggle (Kuznetsov, 2017). The grammatical performance of the model will be trained by checking the output of the model with Grammarly. The semantic performance of the model will be determined by asking end-users to rate the semantic quality of the song lyrics generated. These scores will be compared to avant-garde lyrics. The reason that avant-garde lyrics were chosen is two-fold, one they are unusual and give the RNN a bit of creative licence and two; people taking the user survey or Turing-like tests will be unlikely to have heard of the avant-garde songs. The Turing Test was ostensibly passed in 2014 (Ackerman, 2014) with the model being tested successfully passing for a thirteen-year-old Ukrainian boy named Eugene Goostman; this shows that the bar for what classes as a Turing Test does not always need to be the highest quality or most popular comparison.

4.4 The Website

The website will have an intro page that explains to the users what the purpose of the tests that they will have the opportunity to take. The website will host both of the Turing-like tests and will have a MySQL Database behind it to store the results of the tests.

4.5 The First Turing-like Test

The website will have two tests for the user to take. The first test will show the user a set of song lyrics and ask the user whether they think the song was written by a human being or by a bot. The results of the test will be stored during the session with hidden forms and Get requests. At the end of the test, the results to this point will be stored in a MySQL database so that the performance of the AI-written songs can be analysed when necessary. Queries can then be run on the database to detect which AI-written songs if any successfully passed for human-written songs at least 30% of the time. The website quiz will track and display how many questions the user has gotten right or wrong, but will also reveal how many Turings have occurred. In this case, a Turing will be when a user thought a machine-written song was written by a human being. The data will then be analysed to see whether bot-written songs can pass for human-written songs over 30% of the time in isolation.

4.6 The Second Turing-like Test

The Second Turing-like Test will display two songs to the user, one of which will be a song that has passed the First Turing-like Test if any do. The other song will be a set of avant-garde song lyrics, preferably a set of avant-garde song lyrics that was mistaken for bot-written lyrics quite frequently in the previous test. The two songs will be displayed side by side with buttons underneath each song with the words this one is the bot. The user will click the button corresponding to the song that they think was written by the bot and will then be presented with the next pair of song lyrics in the same format. The results for each user will be stored in a MySQL Database so that the results of this test can be analysed and see if any songs have passed both Turing-like tests.

4.7 The Google Form

A Google form survey will be created that will ask the users to rate song lyrics based on several criteria. They will be asked to rate the songs they see based on the quality of their English, the grammatical correctness, whether the songs have a coherent theme and whether they appear to have been written by a human. Each of these four criteria will have equal weight when determining the overall user score of the song. The sum of the results of these four criteria will be used to determine the final rating for each set of song lyrics. Because this test is relatively rigorous and requires some mental effort, only bot-written songs will be displayed. Each of the criteria will have six different scores that the user taking the survey will use to rate the song lyrics presented. The lowest of those scores will correspond to a score of zero with each score higher representing 5 points higher; this is merely to give the songs an overall rating out of 100, the number 100 was chosen just because of its roundness and because it grants more sensitivity than rating a song out of 10.

4.8 The New Metric

Automated metrics make rating NLG significantly more manageable than putting everything to users for rating, but most automated metrics are complicated and inaccurate at higher levels of NLG quality (Novikova et al. 2017). For this reason, it was decided that a new metric would be produced that would be as simple as possible to implement and would reflect users opinions as accurately as possible and as objectively as possible. The variables that were chosen for evaluating song quality also had to be as objective as possible and as easy to calculate as possible. Either a program had to be made that would calculate the new metric rating automatically or the calculation to devise the rating had to be simple as possible, and the metric score had to be calculated using tools that everyone had available and would know how to use. For these reasons, the percentage of natural English, including slang words was chosen alongside the number of grammatical errors as calculated by Grammarly. The number of words in natural English can be counted manually, or by using a dictionary, after that, it is as simple as plugging the lyrics into Grammarly to get the number of grammatical errors. Grammarly assigns grammatical errors one of two labels, Major and advanced. For this the purposes of this metric, advanced grammatical errors were referred to as minor grammatical errors. It was thought that minor grammatical errors would have a lesser impact on user NLG ratings than major grammatical errors, seeing as minor grammatical errors could include things like slang words. The maximum score for the metric was set at 100, a lower number like 5 or 10 would likely be more accurate, but would lack the sensitivity to objectively measure the quality of songs that might

be similar in quality. The metric has no lower limit, though any NLG that scores negatively is not likely to be rated highly by users or likely to pass for human-written content. The initial formula for the new metric was chosen as (4pE - 2MG -mG - 300). In this formula pE is the percentage of English, MG is Major Grammatical errors, and mG is minor Grammatical errors. All the songs produced by any method will be ranked by the new metric, and a variety of those songs of different ratings will be put to users to rate to determine the accuracy of the metric, the metric will then be refined to fit as tightly with user opinion as possible.

$$y = 4pE - 2G - g - 300$$

Figure 4.1: The Initial Metric Formula

Chapter 5

Development Practices

A significant quantity of IT projects fail before they are even started (Kappelman et al., 2006). This shows that the development practices can be as important if not more important than the code that is written. The best programmer in the world will not get anything done if it is not clear what needs to be done, how it needs to be done or by when it needs to be completed. Therefore it stands that alongside what was done, it is relevant to document how it was done and how it was planned.

5.1 Github

In the decade since its inception, Github has become of the most useful and powerful tools in a software developers arsenal. The ability to quickly and easily go back to a previous iteration of a project is a blessing almost without parallel. Github allows the immediate implementation and documentation every change that is made make and permits rolling back to the previous iteration of a working product if errors are discovered in the build later on. How much this is taken for granted these days is a testament to exactly how incredible this tool is. Without using Version Control Software, all changes made to code would be permanent and not easy to reverse. On top of this, the ability to know what changes were made, when and by whom, whenever is desired is instrumental in keeping track of how well a project is going. Github was used for this project because it works well with JetBrains software, PhpStorm and PyCharm. With a click of a button changes to the code can be committed alongside a brief description of what was changed. All of this means that developing, documenting and testing each functional block of code happen within a remarkably short period. This also combines well with the agile methodology in that you can work on your project iteratively, always aiming for the next deliverable.

5.2 Agile

The agile methodology was chosen for this project because this project was iterative in nature, it lent itself well to the agile methodology. The idea at every step of the way was to produce something that would be submittable in case of disaster or time constraints elsewhere. All progress and the velocity at which that progress was occurring was documented. What would and would not be completed by any particular time was easy to forecast depending on the productivity of the previous weeks. Tasks were prioritised and ordered intending to making sure that the basic framework of every individual deliverable was completed before any particular deliverable was completed in its entirety. The results of this was a series of iterative deliverables that were worked on continuously until the deadline to make sure that every deliverable was completed in a way that was at least submittable far in advance but only really reached its final quality closer to the end of the project. In fact, nearly every deliverable was developed and edited continuously right until the end of the project.

5.3 Pivotal

The tool that was used to document and measure progress as well as to forecast what would and would not be completed within the timeline is called pivotal tracker. It is a tracker that works with the agile methodology and is meant for managing large projects in an agile way. It works as follows, every project and every deliverable in the said project is broken down into its component tasks, and each of these component tasks is assigned a points value, 1, 2 or 3. These points values represent approximately how much effort or how long each of these tasks will take to complete. In this project a 1 point task was something relatively simple that could be achieved in an hour or so, a 2 point task was something that could be completed in a working block of a morning, afternoon or evening and a 3 point task was something that would probably take a whole day. Of course, these point values were estimated earlier in the project, often far in advance of when it would be possible to know exactly how easy or hard, said task would be. Frequently this meant that deliverables were over or underestimated with regards to how hard they would be or how long they would take to complete. However, the purpose of this tool is not really to forecast how hard an individual task will be; it is to predict what can be achieved by when and how long the overall project will take. This is done by calculating the velocity of the project and projecting forward to see what can be completed in the remaining time at that velocity. The velocity was calculated by averaging how many points worth of tasks were completed in the previous three weeks and projecting that value forward as long as it would take to complete the entire project. This somewhat mitigated the inaccurate forecasting

of how long a task would take or how hard a task would be to complete because as long as inaccurate task values were not altered retroactively, tasks that were estimated inaccurately were as likely to be in the past in the project as they were to be in the future. Thus the velocity remained accurate even with inaccurate estimates of task difficulties. By careful and constant monitoring of the projects velocity, it was ensured that the project would be completed before the deadline.

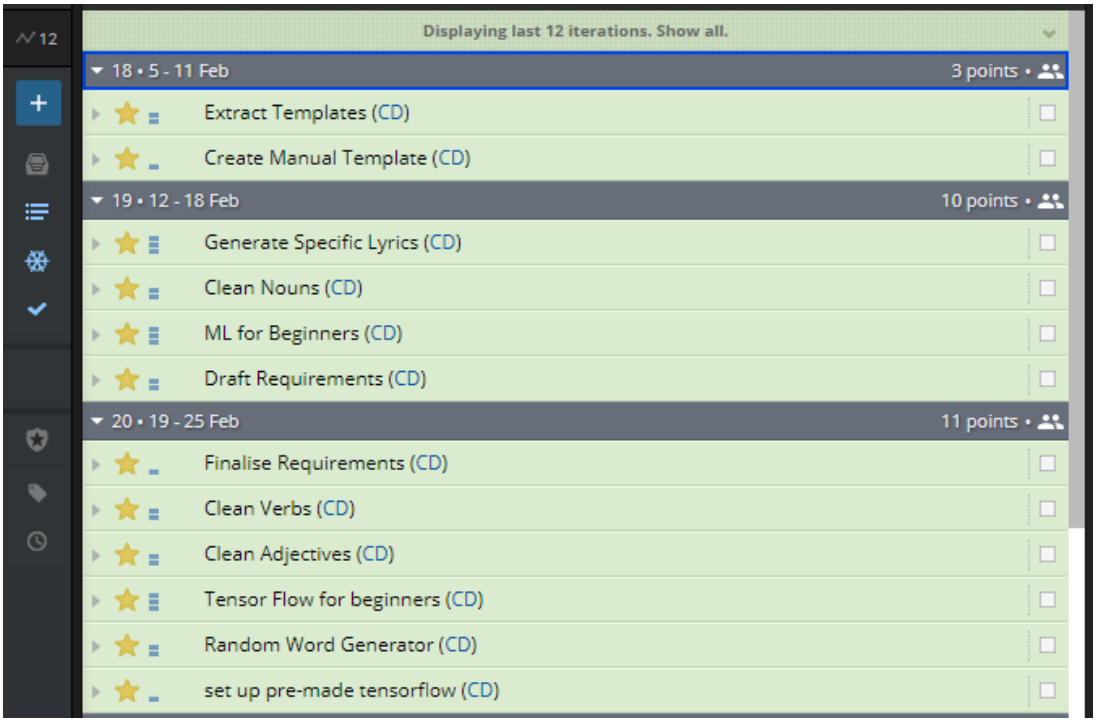


Figure 5.1: Pivotal Tracker

Chapter 6

Implementation

This chapter will document exactly how every one of the systems featured in this project were created. Some systems were changed slightly compared to how they were initially intended to be created but the vast majority of what was implemented was achieved according to specification.

6.1 The Song Lyrics Dataset

When downloaded, the dataset was a 66mb csv and for the largest part, completely unsorted. Before the dataset could be used, the dataset was ordered by artist. Artists that roughly fit the broad genre of classic rock were used to train the model; this was primarily a stylistic choice. However, the RNN proved to be quite susceptible to repetition when it was exposed to it in the training set, and thus, pop music with its focus on repetition would most likely have proven a poor choice of a dataset with which to train the model. Any line or phrase that was repeated more than three times in a row tended to show up absurdly frequently when the RNN was used to generate an original song. Lines that repeated themselves more than twice in a row were removed manually if they revealed themselves by the RNN outputting the same line five or more times in a row. Surprisingly this happened several times while training the RNN. The dataset was ordered by how frequently each word occurred so that the template-based lyric generator could have a bias towards words that occurred more frequently in the dataset. To categorise the words found in the song lyrics dataset accurately into their categories a set of categorised words was downloaded from Ashley-Bovan (Bovan, 2018).

For use in the pattern-based and template-based lyric generators, the Kaggle

dataset required significant sorting. The data was ordered using this relatively simple function.

```
def lyrics():
   counts = defaultdict(int)
   for x in MyFile.read().split():
       counts[x] += 1
   return sorted(counts.items(), reverse=True, key=lambda tup: tup[1])

songout.write('\n'.join('%s %s' % x for x in lyrics()))
```

Listing 6.1: Code to sort the Lyrics Dataset

The function read the lyrics file in its entirety, counted how many times each word occurred and then returned them as a collection ordered from most to least frequently with each word followed by how frequently it occurred in the dataset. The ordered list of words was then printed to a separate file so as not to distort the original dataset.

6.2 The Bovan Dataset

The Bovan Dataset categorised 150,000 words into either nouns, verbs or adjectives. This dataset was further separated into how many syllables each of these words contained. These files were used to automatically categorise each of the words that appeared in the song lyric dataset for use with the template-based lyric generator. However, there were mistakes in the dataset as well as words being listed under multiple categories. The mistakes, as rare as they were, still caused issues considering the size of both datasets and led to the template-based generator frequently using words in ways that they would not be used in most cases. For example, love is a verb in the vast majority of cases, but it can be a noun in some others. This caused some confusion when generating the templates or when running the template-based lyric generator as the lyric generator had no idea in what context the word it discovered was being used. To deal with these issues, a smaller dataset was used for each of the three main categories of words. The Great Noun List (Quintans, 2013) contains a list of roughly four thousand of the most commonly used nouns in the English language. Words from the separate noun files for each syllable length up to four were compared against words from The Great Noun List and only kept if they were in that list; otherwise, they were discarded. This reduced the total number of nouns from ninety thousand including some very obscure nouns and words that could be used as nouns but nearly never were, to just roughly three thousand which were the nouns that were both commonly used in English and commonly used in Song lyrics. The same process was repeated for the adjectives using the TalkEnglish adjectives dataset and then the TalkEnglish

verbs dataset (TalkEnglish, 2017). This reduced the total verbs from roughly thirty thousand each of verbs and adjectives to under one thousand of each and resulted in higher quality lyrics generated with far fewer words misused and uncommon or unusual words appearing significantly less frequently.

This dataset was made more useful using this simple function

```
Cleaned = []
for i in nouns.read().split():
   Cleaned.append(i)

for x in n1.read().split():
   if x in Cleaned:
       clean.write(x + "\n")
```

Listing 6.2: Code to reduce the noun count

What this function did was opened the list of, for example, one syllable nouns and compared it with the shorter more accurate list of nouns, if the word was present in both it was written to another list of one syllable nouns that was now more accurate and contained just nouns.

6.3 The Random Lyric Generator

The random lyric generator created an array of random line lengths, varying from three to ten words representing the length in words of each of the lines of the song that it would create. Once the random lyric generator had at least a total of two hundred in that array, it moved on to the lyric generation phase. At this point for each of the lines of the length described in the lengths array, the random lyric generator chose a word at random from the list of words used in song lyrics in the Kaggle Dataset. This resulted in a song of roughly two hundred words, consisting of lines of between three and ten words.

The lengths of each line were created with this function.

```
Cleaned = []
for i in nouns.read().split():
   Cleaned.append(i)

for x in n1.read().split():
   if x in Cleaned:
       clean.write(x + "\n")
```

Listing 6.3: Creating the line lengths

This created different line lengths that added up to a total of roughly 200 words. Then the lines were populated with words from the dataset.

```
for i in lines:
   line = ""
   for j in range (i):
       number = random.randint(0, len(imported))
       line += " "
       line += imported[number]

   words.append(line)
```

Listing 6.4: Writing random words

6.4 The Template-Based Song Lyric Generator

The template-based lyric generator was made in python. When provided with a template the template-based generator would create an original song according to the rules it was provided. Templates were created by dividing songs into their component parts according to various lengths and categories of word. A one-syllable noun was annotated as 1n a two-syllable verb was annotated as 2v. Any words that could not be categorised as either nouns, verbs or adjectives were simply written in the template as they were. The template-based lyric generator would then choose words at random from the dataset of song lyrics that it was provided with that fit each part of the template. When the generator encountered a word code, it found a word of the correct type from the Kaggle dataset. The word code worked as follows with a number from 1-4 representing the number of syllables in the word and a lowercase letter v, n or a representing a verb, noun or adjective respectively. The generator had a strong bias towards words that occurred more frequently in the Kaggle dataset.

```
for x in f2.read().split():
     #fetch word,freq
   if x in onenouns:
      Templates.write("1n")
   elif x in twonouns:
      Templates.write("2n")
   elif x in threenouns:
      Templates.write("3n")
   elif x in fournouns:
      Templates.write("4n")
   elif x in oneadj:
      Templates.write("1a")
   elif x in twoadj:
      Templates.write("2a")
   elif x in threeadj:
      Templates.write("3a")
   elif x in fouradj:
      Templates.write("4a")
   elif x in oneverb:
      Templates.write("1v")
   elif x in twoverb:
      Templates.write("2v")
   elif x in threeverb:
      Templates.write("3v")
   elif x in fourverb:
      Templates.write("4v")
   else:
      Templates.write(x)
```

Listing 6.5: Writing a template

6.5 The Recurrent Neural Network-Based Song Lyric Generator

The RNN was based on a pre-existing NLG system that was used to write original plays in the style of William Shakespeare (Kofler, 2017). The RNN was trained on a dataset that was downloaded from Kaggle (Kuznetsov, 2017). The dataset was too substantial to economically and recursively train the model, and thus the dataset was broken down into significantly smaller training sets to train the RNN on songs that, although not quite so similar thematically, at least belonged to roughly the same genre. The training set and batch sizes were both increased incrementally until the RNN was able to write original song lyrics in 98+% English making the minimum possible number of grammatical errors, with the number of grammatical errors in

the song being tested with Grammarly. At this point the training set was 2mb, it was making 5 major grammatical errors and 10 minor grammatical errors. The RNN, in general, performed better the more substantial the dataset on which it was trained, the larger the batch size and with more hidden layers than the network was initially designed to have. However, increasing the dataset and number of hidden layers significantly increased the time it took the network to be trained and there were significant diminishing returns once the dataset was over 1mb. The RNN was coded in python, using the tensorflow and keras libraries. It is a versatile natural language generating RNN that can be quite easily adapted to other tasks when provided with appropriate training data. At the end of each session of training, the RNN in its fully trained state was saved as a checkpoint file, to keep a record of how the RNN was after every training session. After each training set, five songs were generated using the RNN testing program. The testing program created a dataset of a specified length from the trained state of the neural network. For testing purposes the length of the generated song was set to 1000 characters, this figure was not chosen arbitrarily, it corresponds to roughly 200 words which is approximately the length of most songs. Producing a set of song lyrics within a character limit meant that lines and words started and ended in the middle of a line, so incomplete words and lines were removed before evaluation. The generated songs were put into Grammarly which told us the number of major and minor grammatical errors. The songs were also evaluated manually by counting the percentage of words that were written in natural English. At each training state, three songs were generated and evaluated, and the best one kept with best being determined by which song had the least grammatical mistakes and highest percentage of natural English words. When evaluating the songs English language percentage was weighted as the most crucial factor with every percentage point of non-English words being equivalent to two major grammatical errors, each major grammatical error was in turn weighted to be equivalent to two minor grammatical errors. This gave each song an overall error score that was ultimately used to evaluate the quality of the generated song lyrics and is referred to as The New Metric.

After the RNN reached peak economical performance being trained on a CPU, the RNN was trained further using a GPU. Training was first performed under the same circumstances as it was when training on a CPU and resulted in better performance under the same conditions but more importantly, if everything else was equal, the GPU trained the RNN roughly thirty times faster. This meant that significantly more iterations could be performed and the RNN could be trained with significantly more massive datasets. Under the conditions in which the CPU trained the RNN to produce 98% English the RNN trained by the GPU was consistently writing in 100% English.

At first glance the song lyrics it was producing looked a lot more like real song lyrics than the lyrics produced by the RNN that was trained by the CPU. Increasing the size of the dataset to 4mb also resulted in the RNN often no longer producing any major grammatical errors and thus the scores of the GPU-trained RNNs significantly outperformed the CPU-trained RNNs according to the new metric.

Having the GPU training the RNN roughly thirty times as fast as before meant that the variables of the model could be iteratively tested to train the model most efficiently and generate the best possible songs. The variables that were tested were, the number of training epochs, the total size of the data that was used to train the model, the total size of the data that was used to validate the model, the sequence length of the data used to train the model, the batch size of the data that was used to train the model, the number of hidden layers of the network, the learning rate and the dropout of the network. Whether the network was trained using a CPU or a GPU could also be considered a variable, but it is simply not economical to test the network further using a CPU to train it. These repeated training sessions with many variables also formed a mini-regression analysis, keeping all variables constant and only changing them one at a time to gradually isolate the effects of each variable on the accuracy and potency of the final neural network. Unfortunately, even with the GPU, the network often took an hour or more to train and thus the data gathered is not enough to be used for accurate regression analysis. With significantly more time or with a significantly more powerful GPU, this could be an area for future research.

It is likely that for most of these variables when comparing their utility and their value that the distribution will be a bell-curve. Unfortunately, the number of tests required to work out the absolute optimum value for each variable would be computationally infeasible and for some variables could even damage the hardware used to train the neural network if said values were too high. To work around how computationally expensive these experiments were, the variables that could be altered without significantly increasing the time that it took the network to train were altered both above and below their base values. Decreasing by each variable by fifty percent and increasing them by one hundred percent to determine which values of which variables resulted in the best performing model as often as possible. A table containing the values of all the variables for all of the trained models as well as how long they took to train and how well they performed is listed in the Appendices. This issue was compounded when predictably the two variables that most increased the accuracy and potency of the neural network were the number of training epochs that the network went through and the size of the training dataset. In fact from the data available, these two variables seemed to have a positive feedback effect on each other, in that they both performed

significantly better if they were both increased. However, both of these variables significantly increased the time it took for the neural network to finish training due to their computationally expensive nature. This resulted in a significantly reduced capacity to run further training cases to work out the approximate value of these variables and the other computationally cheaper variables.

6.6 The User Survey

To analyse how well the new metric had been created and calibrated a user survey was created in Google Forms (see Appendices). This form showed the user five songs of varying quality according to the new metric. The users were asked to rate the songs by four different criteria with six possible answers for each criterion, the choice of six options for rating each of the criteria was chosen for two reasons. First of all six options means that people are forced to take a side, whether they think something is high or low quality. If there are an odd number of options like a standard Likert scale, users tend to choose the middle option (Kalton et al., 1980). Second, the new metric rates AI-generated songs out of one hundred, four criteria would, therefore, be worth a maximum of twenty-five points apiece. With the option for a user to effectively score a song a zero in that category, it meant that the users had to be given six options rather than five. Only five songs were chosen to be rated after initial user feedback said that the test took too long to complete, as of this writing this initial user feedback seems to be confirmed as even in this shortened survey, thirteen people have rated the first three songs and only eleven have rated the final two.

6.7 The Website

The website itself was written from a bootstrap template on a Linux Virtual Machine with a PHP and MySQL back-end. This was done so the website could be created to a standard where it could be put to users to take the tests that the website hosted as quickly as possible.

6.8 The First Turing-like Test

When creating the Turing-like tests, the word bot was used colloquially as the word to refer to any machine-based methods of generating song lyrics. The first test showed the user a song and asked them to guess whether the song was written by a human or by an AI, twelve songs were chosen to be demonstrated. A coin toss was used

to determine whether each song in order would be an AI-written song or a human-written song. Eight AI-written songs of increasing quality according to the new metric featured above were chosen alongside four human-written avant-garde songs. The human-written songs were selected to be relatively obscure and written by avant-garde musicians to make sure that the test takers would be as unlikely as possible to recognise any of the songs. Seven of the Eight AI-written songs were written by this models RNN and one was written by the RNN created by Ghazvininejad (Ghazvininejad et al., 2016). This song was chosen because of all the AI-written poetry or songs featured in the literature above or produced by any of the methods featured in this project; this song was the highest by the new metric. This song was the only song found that scored a perfect 100 and according to the new metric giving it the highest chance of any AI-written song that has been measured by the new metric, of passing a Turing-like Test. The users were shown the entirety of the lyrics to whichever song was being tested at that moment and had to choose one of the two buttons at the side of the screen that they thought represented the correct answer as to whether a bot or human being wrote this set of song lyrics. The buttons were labelled human and bot, the question that was asked of the users was Was this song written by a human being or a bot?. At every stage of this test, after the user had made their guess, they were told of the correct answer and of how many they had gotten right so far. The results of each guess were also stored in the database so that later the data could be analysed to determine if any of the sets of song lyrics had passed the Turing Test. The database contained a Turing column that would be given a value of 1 if any song was written by a bot and a user guessed that it was written by a human and a 0 otherwise, this was done for easy sorting and data analysis.

6.9 The Second Turing-like Test

The second test used the results of the first test to determine which of the bot-written songs were most likely to be mistaken for human-written songs and vice versa. The four bot-written songs that were most likely to be mistaken for human-written songs were compared to the four human-written songs that were most likely to be mistaken for bot-written songs in descending order of likelihood. The users were showed these songs in pairs starting with the pair most likely to be confused, then the second most likely and so on. The users were asked to select which of two excerpts of displayed songs were written by a bot. This test was designed in a way that would make it as likely as possible in the given circumstances that users would select the human-written song. This gave the best possible chance of any of the songs tested passing the Turing Test. The reason that only excerpts were shown rather than full songs was two-fold, this meant that I could get the same pool of users who took the first test to take the

second test and it meant that the songs appeared to be the same length beside each other for purely aesthetic purposes. Once again when getting unpaid users to take tests such as this, it is best to make the test-taking process as painless as possible to ensure the highest level of completion possible.

6.10 The New Metric

The metric was created and used to rate every song that was produced by each of the different methods. Once the results of the first test were in, the metric was re-calibrated, it appears that major and minor grammatical errors affect users perception of song lyric quality roughly equally. So the formula was adjusted to (4pE - 2G - 300) where G now refers to any grammatical errors. The metric was altered by changing the weights of the variables; there were three now there are effectively only two until the loss between the metric ratings and human ratings was as low as possible. Increasing the impact of minor grammatical errors reduced the loss to 114 from 154. In the future, the implementation of the new metric will be iterative; variables will be tweaked until they best match the data thats already available to compare the metric scores with human ratings. New songs will then be rated by the metric, those songs will be evaluated by people, and then the metric will be tweaked again. The ultimate idea is that the metric will have a significant or at least probabilistic predictive capacity, preferably both in the ability to accurately predict user ratings but also to accurately predict the likelihood of a song passing a Turing-like Test.

Chapter 7

Results

7.1 The Random Lyric Generator

Predictably the random lyric generator created lyrics of inferior quality, scoring as poorly as any other song tested on the new metric due to the number of misspelt words and virtually every line making very little grammatical sense. With words just chosen at random and created in a random order, it was supremely unlikely that anything of note would be accomplished, this was merely a control group to prove that machine-written lyrics could be of significantly higher quality than lyrics that were generated randomly. The creation of this algorithm also reinforced the notion that significant work would need to be done with the Kaggle dataset to use it for song lyric generation.

7.2 The Template-Based Song Lyric Generator

The Template-Based Lyric Generator performed similarly poorly, scoring roughly as poorly as some of the worse songs created by the neural network according to the new metric. In some cases, it even performed as poorly as the Random Lyric Generator. Although words of the same type and number of syllables appeared in the same places as they did in the songs that the templates were based on, the algorithm took neither context nor tense into account when choosing which words to place where. This resulted in song lyrics that jumped back and forth between tenses as well as selecting strange combinations of nouns, verbs and adjectives.

7.3 The Recurrent Neural Network-Based Song Lyric Generator

The Neural Network-Based Lyric Generator performed significantly better than anticipated. After significant tuning of the datasets and of the Neural Network itself, the neural network was able to produce song lyrics that scored as highly as a ninety-six on the new metric. These songs were also rated as highly as 61 out of 100 after averaging 13 user ratings. Songs produced by the neural network were also able to pass both of the different Turing-like Tests with which they were tested. According to the first Turing-like Test, after nine users had completed the test, songs written by this neural network had passed for human as many as five times, the Turing test is defined as fooling a human thirty percent of the time five out of nine is fifty-six percent, this is a resounding success. Two others also passed for human three times out of nine and thus also barely pass the Turing Test. Songs created by this neural network also scored a user score of up to sixty one out of one hundred which corresponded to a new metric score of ninety-one. Three of the four songs that were put to the second Turing test passed, strangely, each of the song pairs was tested a different number of times, this could be due to people having connectivity issues, refreshing the page or going back and answering differently. In any case, one song passed for human seven times out of eighteen, one passed twelve times out of twenty-two and one passed thirteen times out of twenty-five. This performance is significantly better than expected.

Eight variables were altered in an attempt to find the optimum value for each variable. As much as possible, independent variables were only changed one at a time in an effort to isolate their influence on the quality of the end product. Unfortunately, due to the computational cost of running these tests and the lack of ability to fully automate this process, the sample size for the number of tests for each variable is not as high as would be ideal. However, this was never the purpose of this project, in any case, a rough idea of how each variable affects the quality can be deduced from looking at the raw data, and this will be discussed here.

Increasing the number of training epochs appears to enhance the quality of the neural network overall, at the cost of significantly increasing the time it takes to train the network. Three different values for the number of training epochs were tested. The five with the largest number of training epochs, twenty, scored an average of eighty-nine on the new metric and the three with the lowest number of training epochs, five, scored an average of seventy-eight on the new metric. From this admittedly

limited data, it appears that increasing the number of training epochs at least within the ranges described will improve the quality of the songs outputted by the neural network but significantly increase the time it takes for the neural network to train. Once the values of the other variables have been accurately deduced, it would be worth increasing the number of training epochs further to see if the quality continues to improve, but the impact that it has on training time makes it an expensive value with which to experiment. Increasing the number of training epochs increased the time it took for the neural network to finish training linearly, doubling the number of training epochs approximately doubled the time it took for the neural network to complete its training. However, when the time taken to train the network is compared to quality, there are significant diminishing returns, quadrupling how long it took for the neural network to train generated songs that averaged a quality 10 points higher according to the new metric. The graph of metric score vs the number of training epochs has a clear line of best fit.

The other variable that had a significant impact on both song quality and training time was the total size of the training set, fourteen different values for the size of the training set were tested and broadly, increasing the size of the training set produced higher quality songs. The songs with the smallest training dataset were unambiguously the lowest quality, but song quality appeared to peak between two and five megabytes of training data. It is impossible to tell from the data produced whether this peak was inherently due to these values being optimum independent of other variables, dependent on other variables or whether the quality of the dataset at that value just happened to be superior. Unfortunately, due to the significant number of songs that went into making up datasets that size, precisely which songs were included in which datasets were not recorded, 5mb of song lyrics corresponds to roughly 9000 songs. Even if all the songs were recorded for each dataset, working out the effects of any of the individual songs would be nearly impossible from this data. Increasing the size of the training dataset in isolation always increased the time it took for the neural network to train linearly, doubling the size of the dataset roughly doubled the time it took for the neural network to finish training. The graph of training dataset size vs new metric score is a rough bell curve rising in quality sharply as the dataset increased before falling off more gradually after the peak training set size was hit.

The sequence length of the data used to train the neural network was only successfully changed twice with no other variables changed alongside it, decreasing the sequence length by a factor of 5 resulted in a song of lower quality and a neural network that took significantly longer to train. Increasing the sequence length from 50 to 75

resulted in decreased quality but a faster-trained network, increasing the sequence length beyond 75 resulted in a hardware fault. Perhaps superior hardware could be used to experiment with the sequence length, but it does not seem like there will be significant improvements beyond 50 from this admittedly limited data.

The batch size was the most interesting variable that was changed. When using the CPU, increasing the batch size improved both the quality of the songs generated by the RNN and the time it took for the neural network to train, but at the cost of extra strain on the hardware. When altering batch size while training the neural network with a GPU, batch size seemed to result in optimum lyric quality at 256; this is probably once again a result of hardware limitations, a superior GPU could maybe have handled a larger batch size. Curiously, doubling the batch size did not half the time it took for the network to train given identical conditions, it reduced the time to train by one third.

Increasing the number of layers of the network from 3 to 4 resulted in a noticeable increase in quality under similar conditions. Increasing the size again to 5 layers resulted in a slight decrease in quality and an increase in the time that it took for the network to train, for this reason, the number of layers in the network was not tested above five layers, once again the differences in quality between the different numbers of layers was relatively small so its hard to determine from the data what number of layers would allow the RNN to perform the best.

The learning rate was increased twice to increase the rate at which the neural network could become fully trained. However, in one of the two cases that it was tested at the doubled rate from 0.001 to 0.002, a network that had been performing admirably suddenly produced by far the worst sets of song lyrics that had been seen from any of the networks in at least the previous dozen iterations of training. The decrease in quality on increasing the learning rate was probably because the neural network overwrote itself too significantly when training itself on the character it had seen most frequently. This hypothesis is backed up by the fact that the song lyrics produced after increasing the learning rate seemed to contain a disproportionate quantity of the letters K, E and N Reducing the learning rate from 0.001 to 0.0005 also resulted in a slight dip in quality. It could be because the network had not quite had time to reach an optimum trained state yet, but it is not computationally worth increasing the time it takes the network to train without ensuring that it will improve song lyric quality. With a limited computational capability to run repeated tests, this variable took a backseat to variables that apparently had a more significant effect on generated song

lyric quality.

The dropout was only changed one time, it was increased very marginally from 0.8 to 0.85, and it resulted in the neural network taking over 20% longer to train while producing songs of the same quality that it was creating before. For the same reason as with the learning rate, no more experiments were performed on this variable so it is impossible to discern where the optimum value might lay from this limited dataset.

7.4 The First Turing-like Test

Of the songs written by RNNs, four of them passed for human over 30% of the time; this included the three songs highest rated by the new metric. Three of the four human-written songs passed for bot-written songs over 30% of the time with one being mistaken for a bot-written song over 65% of the time. This could either mean that users were guessing bot more often than they really thought because they did not want to be tricked or that the avant-garde songs that were chosen looked how users would imagine bot-written songs to be written. The songs rating with the new metric correlated strongly with how likely a song was to pass for human with the top three songs passing more than 30% of the time and none of the bottom three passing over 20% of the time. Dropout was roughly 20%, with 14 people answering the first of 12 questions and 11 answering the 12th.

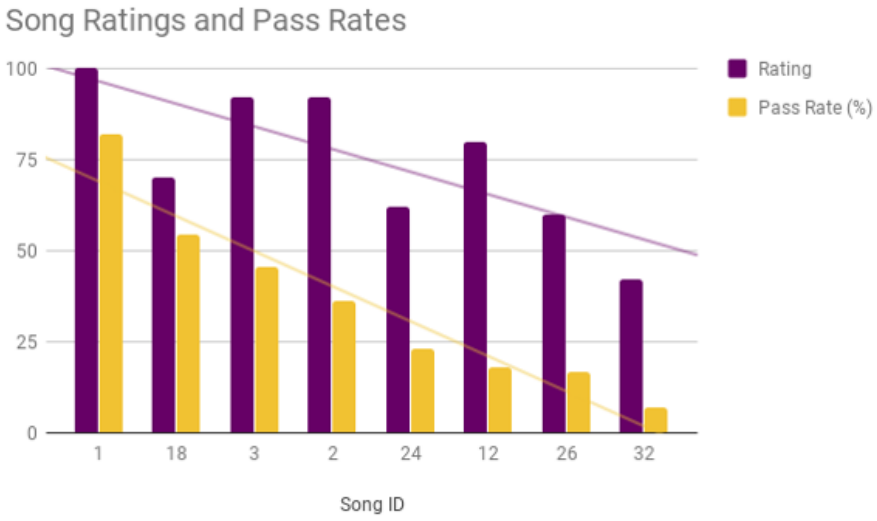


Figure 7.1: Results of the First Turing-like Test

7.5 The Second Turing-like Test

Three of the four bot-written songs passed for human over 30% of the time in the second test. All three of the songs that passed the second test were written by the RNN-based song lyric generator. All of the human-written songs passed for human over 50% of the time, showing that when the decision was more adversarial, people were more likely to recognise human-written songs even if they were avant-garde. Interestingly, drop-out in this test was extremely low, 62 people completed the first of the four questions and 58 completed the fourth and final, this shows that this testing method would work in the future with significantly larger sample sizes.

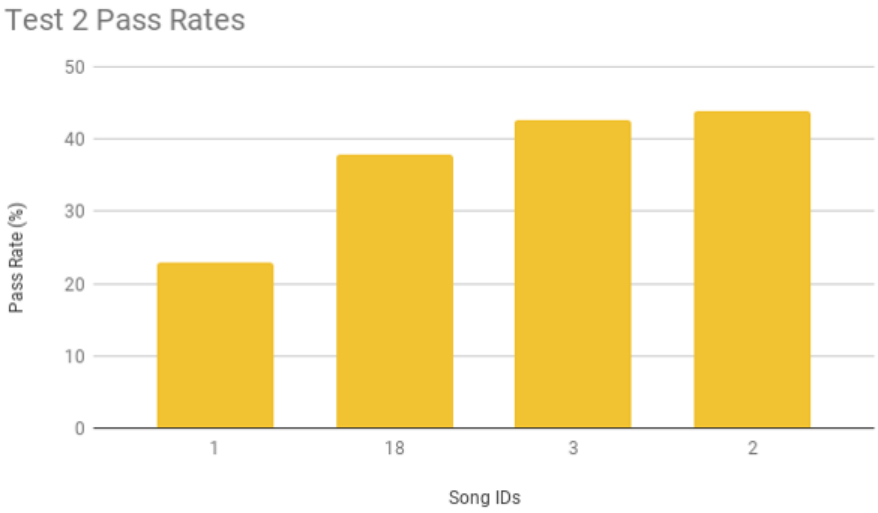


Figure 7.2: Results of the Second Turing-like Test

7.6 The New Metric

The results from the Google Form survey were used to calibrate the metric. The order of the ratings from the new metric exactly matched the order of the human quality ratings when taking into account that two of the five songs scored the same score out of 100 when put to user evaluation. Like other metrics, the new metric more accurately reflected human opinion at the lower end of the rating scale and performed somewhat worse at the higher end of the scale (Novikova et al., 2017). Before recalibration, the new metric was out by just 3 points on the poorest of the lowest ranked of the five songs, but out by a full 28 points on the highest rated of the five songs. Before calibration, in total the metric was out by 154 points.

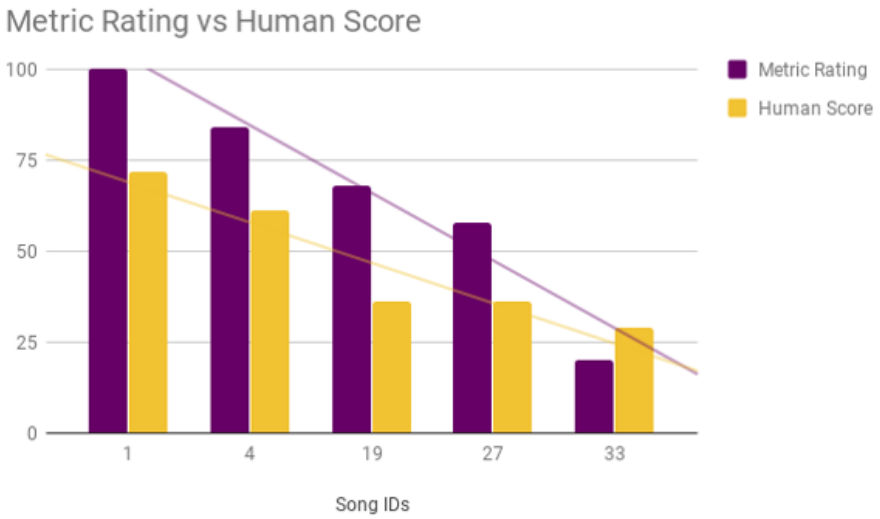


Figure 7.3: Google Survey Results

In all five cases, the metric overestimated human opinion and thus during recalibration, the idea was to make the metric rate songs more harshly, and therefore it was decided that minor grammatical errors would have the same effect on the rating of songs as major grammatical errors.

$$y = 4pE - 2g - 300$$

Figure 7.4: Recalibrated Algorithm

After this recalibration the metric more closely ranked 3 of the five songs and the total discrepancy between the scores of the metric and the user ratings was down to 114. After recalibration, the same problem with accuracy at lower rankings and less accuracy at higher rankings remained. The metric was now out by 9 points for the lowest-rated song and still out by 28 points for the highest-rated song. The metric also shows utility in that songs that are higher rated by the metric have increased chances of passing for human-written songs. Every song that scored over 90 on the new metric passed for human over 30% of the time on the first Turing-like test, conversely, none of the songs that scored under 70 passed for human over 30% of the time. This shows that unlike in reflecting the overall human opinion, the metric shows utility in estimating how likely a song is to be passed for human both at the upper and lower score ranges. Out of the eight songs that were tested, all of the top three passed for human over 30% of the time and none of the bottom three. However, the fact that a song that scored the perfect maximum score of 100 when rated by the new metric and only 72 when put to user ratings shows that the metric is not taking into account every possible variable

that humans, either consciously or unconsciously, use to rate songs. More research is needed to calibrate the metric further, taking into account other variables that can be objectively measured. The issue once again is objective measurement, it is virtually impossible to rate the semantic-coherency of a song objectively, but it is possible to count how frequently words are repeated, or how many lines have the same number of syllables or how many lines in a song rhyme.

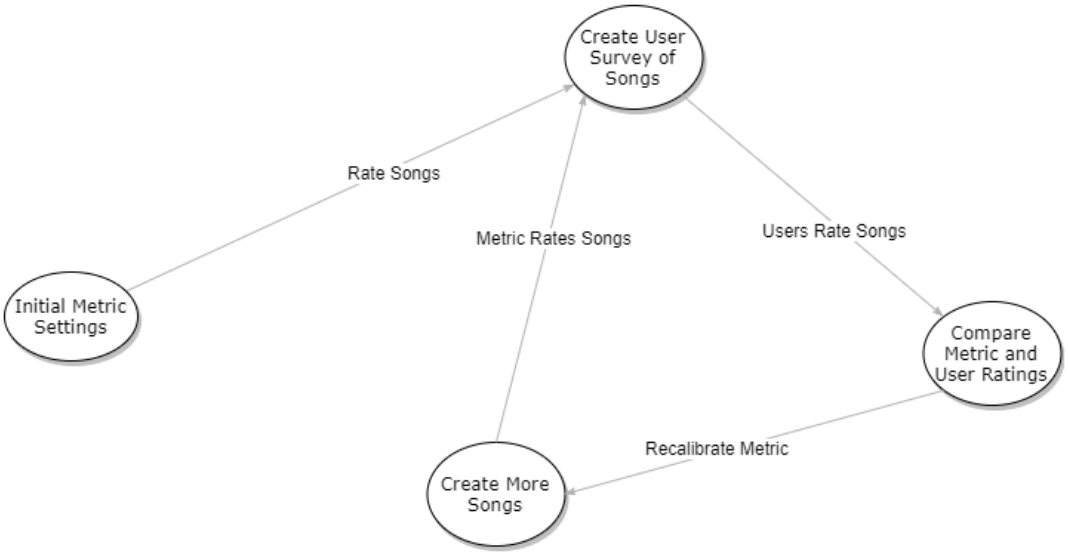


Figure 7.5: Recalibrating the Algorithm

Chapter 8

Limitations and Evaluation

This chapter evaluates the quality of the work produces and makes suggestions as to what could be improved in the future or could have been done better in the past.

8.1 The Kaggle Dataset

This dataset was scraped from the Lyrics Freak website which is a website where users submit what they think are the lyrics to various songs and then those sets of lyrics are rated by the other users. The lyrics that are the highest ranked are the lyrics that remain on the site; this means that lyrics to more obscure songs are less likely to have been significantly edited. Thus they will likely have more mistakes. When the data was scraped, no consideration was given to the relative obscurity of the song lyrics scraped. There are significant numbers of poorly written words, and there are likely to be misheard lyrics in this dataset, this is evidenced by looking at the ordered list of all the words occurring in the dataset. Many of the least frequently occurring words are obvious typos. The dataset was not sorted by genre which made it harder to train an AI that would make a coherent song. Songs of different genres are vastly different which meant that the dataset had to be sorted manually to create a dataset that an RNN could use to have any chance of creating a song with a coherent theme.

The dataset could also have done with stemming and lemmatisation, primarily when working with the template-based lyric generator, this would have given a more accurate idea of which words were used most frequently as well as ensuring that the same words were used in their same forms consistently.

This is not necessarily a problem with the dataset per se, but many of the songs in

this dataset contained excessive repetition, one of the issues with RNN-based NLG systems is that they tend to repeat repeated phrases excessively. This repetitiveness is because their output becomes part of their next input and thus if the input and output are the same this can be a hard cycle to break. To prevent this from happening the dataset would need to be parsed to find repeated strings and then have excessively repeated strings removed.

8.2 The Bovan Dataset

The Bovan dataset contained many mistakes, but even in cases where it was right, it was not necessarily useful, many words fell under multiple categories and had to be separated out into their more frequent uses manually. Words also did not fall under a pronoun category, considering that many of the most frequently used words in song lyrics are pronouns, this is a significant issue when creating song lyrics from this dataset.

8.3 The Random Lyric Generator

The random lyric generator did not perform well, but in reality, it was never meant to, it was merely created to demonstrate that other machine-based methods could be significantly more effective than a random word generator. However, with a superior dataset, it could still have been used to create songs of slightly higher quality.

8.4 The Template-Based Song Lyric Generator

The Template-Based generator did not prove as successful as was hoped, although words had roughly the correct number of syllables, the template had no idea of semantics, rhyme, rhythm or tense. Words of past and future tense were put next to each other and combinations of adjectives and nouns appeared that made no particular sense. These issues could be resolved, but even then it is unlikely that a solely template-based method of generating lyrics could ever be as proficient as one that incorporated neural networks (Oliveira,2009)(Ghazvininejad,2016). The template-based model also merely mimicked the template that it was assigned, it could have been used to find the average template of all the songs on the Kaggle dataset and then used to create an average song, whether this would be better or worse would require further research.

8.5 The Recurrent Neural Network-Based Song Lyric Generator

Although the RNN was able to produce songs that passed Turing-like tests, the best song it produced that was put to users for evaluation only scored 61 out of 100, that leaves a lot of room for improvement. The most significant issue with training the RNN was that it takes a long time, and the computational resources required mean that when training an RNN little else can be done on the same machine. This meant that significantly fewer iterations of training and adjustments of variables could be performed than would be required to work out the exact combination of variables that would work best. Even from the 35 iterations of training that have now been completed with this dataset, some of which lasting as long as 9 hours, it is still unclear which variables have the most significant positive impacts on the quality of the song lyrics that the RNN-based generator can produce. Also, just five tests were run after each training iteration, it is possible that with more tests, some of the trained states could have generated better songs, but when time to train is already an issue, a line has to be drawn somewhere. It is also possible that some of the trained states generator significantly better songs than they usually would have, seeing as only the best of the five was taken and recorded, the other four could have been considerably poorer and a better representation of what the RNN would be more likely to produce in that state. The RNN was, barring two occasions, trained mainly on the genre of classic rock, it is possible that other genres could have been used to train the RNN to produce better songs. Ultimately, the most prominent limitation of the RNN was the time that it took to train.

8.6 The Google Form Survey

The Google form survey that the metric was calibrated on only featured five songs. Roughly 20% of the users who took this test dropped out after just three songs, so, unfortunately, it would be quite challenging to get a significant number of users to rate significantly more songs accurately in a single test. It would be more plausible to create different five song surveys and send them out to different groups of users. However, there were not enough sufficiently large pools of unique users available to get to take the test. This survey and both of the Turing-like tests had overlap in the songs that were featured. This was necessary to work out the correlations between user-ratings, the new metric rating and the likelihood of passing both tests. The Google Form Survey also only featured four categories on which to rate the songs, these four categories were chosen based on the categories that were used to assess other NLG systems (Oliveira,

2007, 2009) but were ultimately arbitrary and could very well have missed the essential qualities that make a song human-like. Another survey could have been performed to determine which categories to rate songs by, but would listeners even know what the factors were that caused them to like songs that they like? The literature seems to suggest otherwise (Schachter and Singer, 1962), people tend to create reasons for their opinions after the fact rather than approaching things with their criteria for enjoyment in mind in advance and evaluating whether those criteria have been met afterwards. More potential factors for what might make a song seem human-like to readers or listeners need be proposed and evaluated. The four categories that were used to assess human ratings of the song lyrics were also very strongly correlated with each other. It is possible that the respondents were somewhat lazy and merely assigned songs roughly the same score for each category rather than thinking about these criteria individually, it might be better to have put each of these criteria to users separately so respondents would not just be tempted to assign the same score to each. Although it is not known precisely who the test takers were for this test, this test was only shared with students of the university and specifically within the university computing department. This is an abnormal and specific subset of the general population, younger and more educated than most; this could have influenced the quality of the responses to this test.

8.7 The First Turing-like Test

The human lyrics chosen were all from the avant-garde genre, this was so nobody who took the tests would have been likely to have heard the songs and thus would not know the answer to the question in advance. Avant-garde song lyrics were compared to song lyrics generated by an RNN that was trained mostly on classic rock; those are different genres. It is plausible that one is perceived more human-like than the other and thus it was merely the nature of the genres that made people choose the bot-written songs as frequently as they did. Of the 35 songs that were created by the neural network and several that were created by the pattern-based and random word generator-based systems, only eight were ultimately chosen for the Turing-like test. It is possible that even more human-like songs were left untested and that even after all this testing the song most likely to pass a Turing-like test was not discovered. Dropout for this test was quite low around 20% overall on a 12 question test; this implies that the test could have been significantly longer and more data could have been gained, though it was impossible to know this in advance. Most of the people who took this test had at least a basic knowledge of AI; it could be that users without this experience would have guessed that bot-written songs were written by human beings more frequently. This test was also only taken by 13 people; this was because, once again, the first and second Turing-like tests involved the same bot-written songs and thus could not both

be taken by the same users. There were not enough sufficiently large unique pools of users to get to take the tests.

8.8 The Second Turing-like Test

The second Turing-like test used songs from the previous test to see if they could pass a more rigorous test, but each bot-written song that was chosen from the last test was only compared to one human-written song and vice versa. It could be that these songs only performed the way they did under that specific comparison and not that they would be interpreted similarly when compared to other human-written songs. Only four bot-written songs were tested, this was because only four human-written songs were tested in the previous test, but this is still a small sample size, small even as a proportion of the 35 songs that were created by the RNN-based generator. Only four human-written songs were tested, once again this was just because these were the songs that were evaluated in the previous Turing-like test, but this is still a low sample size. Dropout was exceptionally low, 5% after four pairs of songs were tested, this shows that more song pairs could have been tested without the worry that a significant proportion of the users would not complete the test. The test was designed with the view that it was unlikely that any of the bot-written songs would pass for human-written songs over 30% of the time. To this end, the bot-written song most likely to be confused for a human-written song was compared to the human-written song most likely to be confused for a bot-written song, with the second most confused pairs being tested and so on. Despite this, the pair that was predicted to be the most likely to be confused were the only pairing in which the bot-written song was confused for being a human-written song less than 30% of the time. In any case, it may have been more productive to pair the human-written and bot-written song randomly. Each pair of the songs excerpts that were compared in the second Turing-like test were artificially made to be the same length. Whichever of the two songs was longer was cut in length until it had the same number of lines as the song to which it was being compared. This was for aesthetic purposes so that the buttons the user would use to select what they thought was the bot-written song were right next to each other and on the screen without the user having the scroll. The lengths were also made to be the same so that users could see the entirety of both songs on the screen at one time. It is unlikely that this problem could have been solved in a way that would still have resulted in such a low dropout rate from this test.

8.9 The New Metric

When calculating an objective score for each of the songs, minor grammatical errors counted against the possible score for the song, however many songs are deliberately written using slang words like "gotta"' or "shoulda"'. The inclusion of slang words such as these is unlikely to have a significant adverse effect on the perception of any song containing, but excepting slang from the pre-existing class of minor grammatical error would add a manual layer of complexity to what is designed to be a simple algorithm. As expected, the dataset with which the model was trained contains a significant number of slang words and thus according to this new metric many of the songs will have lower ratings than they possibly should. The initial weighting of the metric itself is also, admittedly, completely arbitrary, but the weightings for the three initial criteria had to start somewhere. After analysing the results of the first 13 user surveys, the automated metric appears to fall into the same trap as other automated metrics in that the metric accurately represents user opinions at the lower end of the metric spectrum and becomes less accurate at the higher ends of the spectrum. A song that scored 26 on the automated metric scored 29 on average when scored by users, but a song that scored a perfect 100 on the metric only scored 72 when put to end users. The fact that the song scored a perfect 100 and was not rated anywhere near that highly by users also shows that it is not merely a case of tuning the metric to make it more accurate. There have to be additional criteria that we measure to determine how a user rates the song that we have not taken into consideration. Whatever these other criteria may be, it could be hard to rate it objectively. It seems like a Neural Network might be the best tool for creating an automated metric, but it would need training with a vast and diverse dataset of songs and user ratings. It is also unclear at this point what the metric is best at measuring, is it measuring how likely it is that a reader will think that the song will be confused for a song written by a human or is it merely evaluating the overall perceived quality of the song? It seems likely that those two things would be correlated, but at this point the answer to the question is uncertain. The only bot-written song that was tested that did not pass for being human-written over 30% of the time when compared adversarially to a human written song was the song that was highest rated by the metric overall. That could merely have been the result of the song that it was paired with, even though that song was the song most likely to be confused with a bot-written song in the previous test. This song was also the song highest rated by the Google Form Survey responders, so perhaps it merely demonstrates that quality and the perceived likelihood that a song was written by a human are not the same things.

At the moment, the metric penalises songs for their absolute rather than relative

numbers of grammatical errors. By sheer probability, longer songs would be likely to have more grammatical errors, and thus the metric would need fine tuning if it was to be used for longer songs. The metric as it stands only works as well as it does because all of the songs it has been used to rate were roughly the same length.

8.10 Overall Evaluation

Ultimately, the most significant issues with this project were all related to sample size. Not enough songs were created by the RNN to determine the impact of the various RNN variables, not enough songs were put to users to rate in either the Google Form Survey or the first Turing-like test. There also were not enough unique groups of people that the Google Form Survey or either of the Turing-like tests in the way that they were designed. If there was less overlap between the survey and the tests then overlap between the groups who took the survey and the tests would not be such a big deal, but the nature of the tests meant they had to be taken by unique groups of users. Conversely, if the survey and the tests had less overlap in the songs that were tested, less information would have been gained about how well the scores from the New Metric correlate to how likely songs are to pass either of the Turing-like tests. Further, many of these problems are mutually exclusive, in that you cannot prevent one without causing the other.

There were issues with the samples of users to which the survey and Turing-like tests were put. The Google Survey was more or less only put to university students aged 21-35 who also all had some training in Artificial Intelligence, this isnt really representative of the general population, but it was unavoidable given the number of different groups that were needed to take each of the tests. The first Turing-like test was also only put to a sample very similar to the Google Form Survey. The second Turing-like test was shared via Facebook, but the group that it was shared with share some significant similarities with the groups that took the first Turing-like test. Although no details about any of the users were recorded, its safe to say that those who took the second Turing-like test probably had an average of a college or university education and were nearly all aged under 30, this is once again not an accurate sample of the general population.

The main conclusions for this chapter.

Chapter 9

Conclusion

9.1 Conclusions

This chapter will evaluate the extent to which this project was successful. The benchmarks for success in this project were whether a lyric generation system could be created that would pass create lyrics that pass for human-written lyrics over 30% of the time. To create a new metric that accurately represents human opinion. For the new metric to accurately represent the chance for song lyrics to pass for human-written lyrics. To that end, all three of this principle objectives were completed, though the metric still needs significant calibration. This chapter will conclude with a less-formal evaluation of how this project, and perhaps just as importantly, how I, performed.

9.1.1 Details

Datasets exist that after some altering can be used to train Recurrent Neural Networks (RNNs) to write song lyrics. A relatively simple RNN can be used to generate original song lyrics that pass for human-written lyrics up to 81% of the time when displayed on their own. The bot-written song lyrics can also pass for human-written lyrics up to 45% of the time when displayed adversarially alongside existing human-written avant-garde song lyrics in a test where the users have to choose which of two sets of song lyrics was written by an Artificial Intelligence (AI). A new simple to use automated metric for measuring NLG was created. When this metric is used to rate song lyrics, it appears to match user opinion reasonably well, with a difference of 8 points out of 100 between the metric and user opinion at lower levels of quality and a difference of 28 points out of 100 at higher levels of quality. This new metric also seems to correlate quite well with how likely a set of song lyrics is to pass for a set of human-written song

lyrics. Three of the four highest-rated sets of song lyrics tested passed for human-written lyrics over 30% of the time as well as none of the lowest rated three. No song rated under 70 passed for human-written over 30% of the time when shown in isolation and no song rated over 90 failed to do so. With new more straightforward automated metrics and new superior datasets, modern RNNs can be used to generate high-quality song lyrics efficiently and effectively.

9.2 Future Work

This project leaves plenty of room for future work. Both the lyric generation and quality measuring systems could be improved and leave plenty of opportunities for a lot of fine-tuning.

The goal of the RNN was to create songs that would pass for human-written songs. The goal of the people in both Turing-like tests was to guess which songs were written by bots. Both of these tasks could have been accomplished simultaneously by an Adversarial Neural Network (ANN) (Goodfellow et al. 2014). An adversarial neural network is a system consisting of two separate neural networks competing with each other to accomplish opposing competitive tasks. In this case, one neural network would be trying to write human-like songs, and the other would be trying to guess whether those songs were written by a human or a bot. This system could prove challenging to set up as actual human-written songs would have to be interspaced with bot-written songs to prevent the neural network responsible for guessing which songs were written by human beings from just guessing bot every time. Hypothetically with this system both neural networks would be fine-tuned automatically and tests would continue to run until neither network was appearing to improve anymore. This system might prove useful, but it would be incredibly computationally expensive and likely be very time-consuming. Unlike with other forms of neural networks, there would be no way of knowing in advance how long the system would take to reach equilibrium.

Once further measurable factors to include in the new metric have been deduced, a neural network could be used to work out the appropriate weights of those factors. This would require an extensive dataset of human-rated songs, but could test all of the possible values of all of the possible factors repeatedly until it finds the combination that most accurately represents human opinion. This would also likely prove quite computationally expensive, and it would likely take a significant period to produce a large enough dataset. This solution, however, would likely prove less technically challenging than an ANN.

The dataset mentioned above would need to be generated in advance. A large bank of neural network written songs would need to be created and put to users for rating. This would need to be done in a way that was entertaining otherwise dropout would be too high to get sufficient results. Alternatively, Amazon Mechanical Turk could be used to recruit users for a nominal fee, paid users are likely to be liable to take less entertaining tests.

Once the new metric has been refined further, a neural network to manage the RNNs variable values could be created; this neural network would alter the RNNs variables to try and score as highly on the new metric as possible and as consistently as possible. Once again, this solution would prove exceptionally computationally expensive, but it could prove to be very effective.

More efficient datasets with which to train neural networks to write songs need to be created, perhaps consisting of more songs, songs lyrics with less spelling mistakes or songs sorted by genre, all of these factors would need to be tested. Training an RNN with these assorted conditions could be used to determine which combination of training set factors can produce the highest quality sets of song lyrics. A neural network in charge of selecting the dataset with which to train the RNN could be used for this task too, but once again this would prove computationally expensive.

A superior pattern-based method for generating song lyrics needs to be created as the one created by this project did not meet state of the art standards. A hybrid system that uses an RNN to generate songs that follow a specific pattern is a plausible solution that could produce higher quality song lyrics or lyrics more likely to pass for human-written. Numerous types of models could be tested, but the literature provides possible solutions (Ghazvininejad et al. 2016).

Superior datasets for generating songs with a pattern-based lyric generator need to be created; the current existing datasets are not sorted well enough by likely purpose or by tense to develop coherent human-like songs with consistency. Perhaps Monte-Carlo Methods (Kumagai et al. 2016) could be used to create sample sentences of words that are likely to follow each other allowing for the simple, rapid generation of human-like songs with a pattern-based system.

The metric could be tuned further to provide not just a rating of a song written by a

neural network, but with that rating, it could announce the probability that a song would pass for a human-written song, either on its own or when compared to another human-written song. It is likely that even very highly rated songs would not pass for human-written 100% of the time and it could be useful to know just how strongly a high rating correlates with how likely a song is to pass for human-written.

It could also be that the metric score can be tuned to either accurately represent user opinion of the song lyrics or the likelihood that a set of machine-generated song lyrics might pass for human-written lyrics, but that these concepts are not as linked as was hypothesised. If this was the case, a neural network could be used to tune the metric, not to user scores, but to how frequently they pass for human-written lyrics.

Randomly selected songs from a pool of human-written and bot-written songs could be displayed to users side by side, with the user asked to guess which song was written by the human. This would provide a list of songs perceived as most human-like and most bot-like which would give a starting point as to which characteristics were responsible for these perceptions. Once again, Amazon Mechanical Turk could be used for this process.

It is also worth testing the new metric on other similar forms of NLG such as poetry, to see whether it has a diverse array of use cases or is otherwise more esoteric. It seems plausible that with simple changes specific to the form of NLG that is being tested, the metric could prove to have a more diverse array of uses.

9.3 Reflection

In the end, I am happy with what I achieved with this project. If I were to do it again, I would spend more time working on the RNN, start that earlier rather than focussing so much on the pattern-based lyric generation system and I probably would not make the random word generator at all. I would also find a way of acquiring a GPU quicker to train the neural network better. I also wish I had realised in advance how potentially useful the new metric could have been. If I had, I would have spent significantly more time on devising and tuning the metric. In an ideal world, I would have gotten info about the groups of users who did the survey and the tests, I would have done more iterations of the tests and dropout would still be as low as it was, but this is not really feasible. In the real world, the only things I really think I could have done differently would have been to tune the RNN further to generate better songs and possibly test the metric on further iterations of analysis and tuning.

Appendix A

The Website

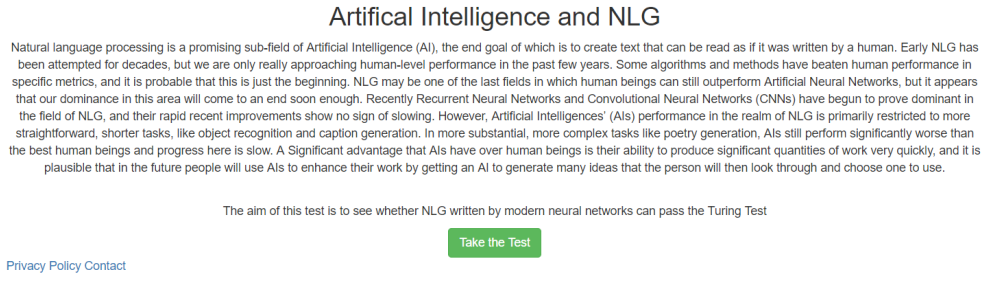


Figure A.1: The Website Intro Page

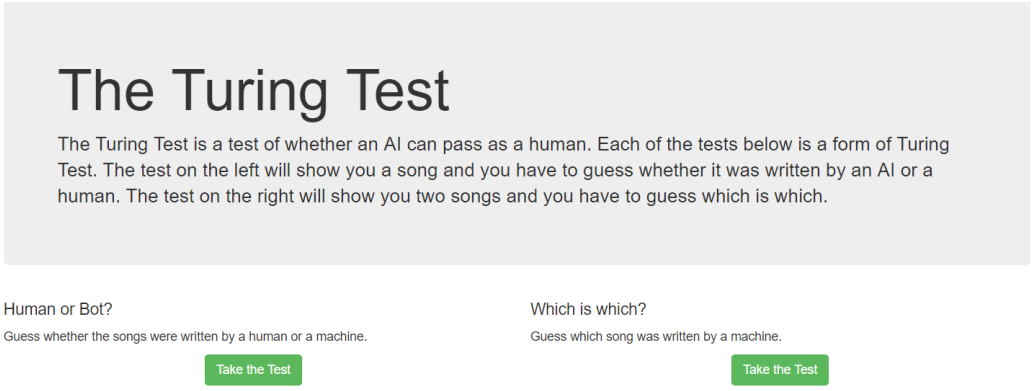


Figure A.2: The Test Intro Page

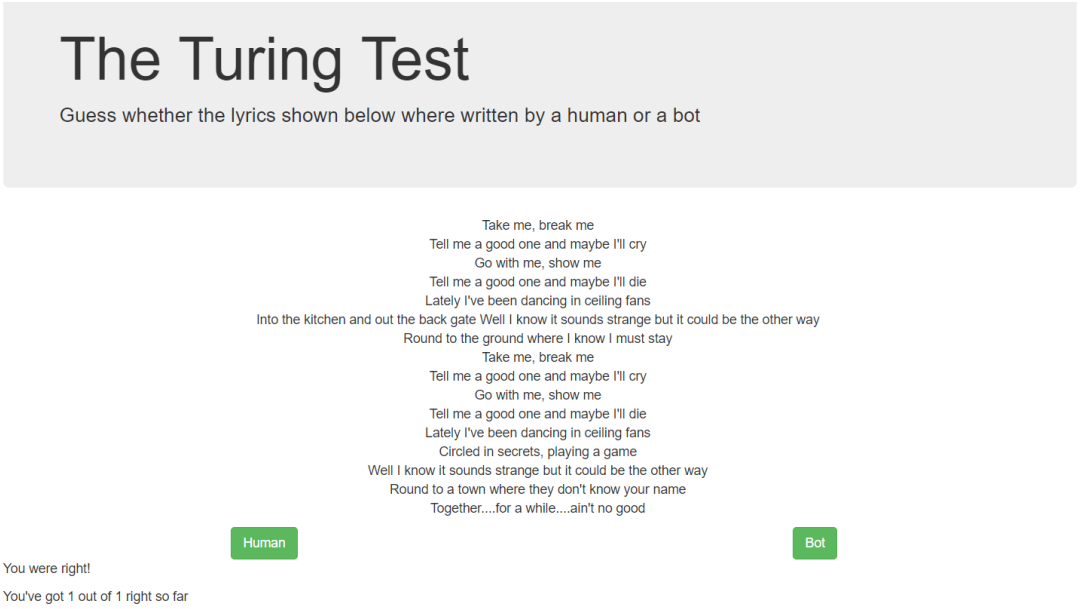


Figure A.3: The First Turing-like Test

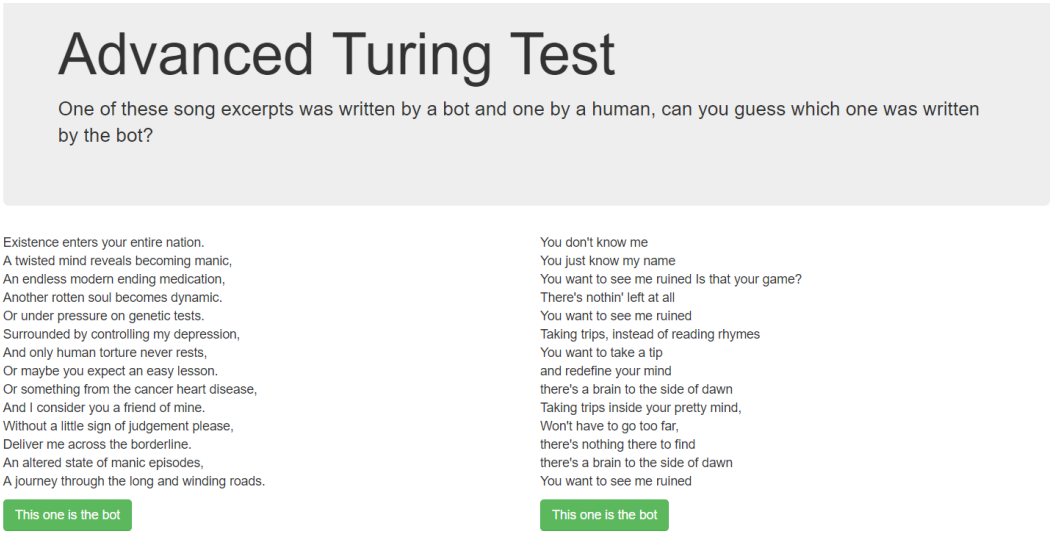


Figure A.4: The Second Turing-like Test

Song Lyric Evaluation

The purpose of this test is to see how people rate human-created and bot-created song lyrics, you will be shown 5 songs and asked to judge them on various listed criteria.

This test should take about 10 minutes to complete.

* Required

Song 1

Please read the song shown below

Existence enters your entire nation.
A twisted mind reveals becoming manic,
An endless modern ending medication,
Another rotten soul becomes dynamic.
Or under pressure on genetic tests.
Surrounded by controlling my depression,
And only human torture never rests,
Or maybe you expect an easy lesson.

Or something from the cancer heart disease,
And I consider you a friend of mine.
Without a little sign of judgement please,
Deliver me across the borderline.
An altered state of manic episodes,
A journey through the long and winding roads.

1. **Please rate the song above on the following Criteria** *
   Mark only one oval per row.

	Atrocious	Poor	Fine	Good	Very good	Excellent
How is the grammar of the song above?	◯	◯	◯	◯	◯	◯
How is the theme of the song above?	◯	◯	◯	◯	◯	◯

2. **Please answer the question below** *
   Mark only one oval per row.

	Strongly disagree	Disagree	Slightly Disagree	Slightly Agree	Agree	Strongly Agree
The song reads like the writer's first language is English	◯	◯	◯	◯	◯	◯
The song reads like it was written by a human	◯	◯	◯	◯	◯	◯

Song 2

Please read the song shown below

Your mondy way of mine, gone
If you care?
Nobody goes where you can't let go
Feel the rest of the bard
Don't be long
Please don't be long

Please don't be long
Please don't you be very sonn

You better get it while it's got a gun (Thinkis a falls away

Lottle window, and right
Please don't be long
Please don't be long
Please don't be saving

Passed my money
And if they call them the Diamond Dogs
Don't be saving around

Poosing on a chance
Got to be your sisce

Blease don't be long
Please don't babe

Bet you took a donet man
But you won't get no but you're ready for more
Ever one says of yourself
You're gonna be a right
If the morning in the streets
I hope you're head of my chance
If you can feel a lottle for me
Gotta have a mother for me
Gotta have, gotta have, gotta have, gotta have,
Gotta have,
Sot a hothen wore
Lottle gizl and have you dance
Oh, haney wat ham
She got a woman and when

3. **Please rate the song above on the following Criteria** *
Mark only one oval per row.

	Atrocious	Poor	Fine	Good	Very good	Excellent
How is the grammar of the song above?	◯	◯	◯	◯	◯	◯
How is the theme of the song above?	◯	◯	◯	◯	◯	◯

4. **Please answer the question below** *

Mark only one oval per row.

	Strongly disagree	Disagree	Slightly Disagree	Slightly Agree	Agree	Strongly Agree
The song reads like the writer's first language is English						
The song reads like it was written by a human						

Song 3

Please read the song shown below

We can't stand a bar fool
And I'm gonna cry for me,
And I'm gonna change my mind
I con't come but I'm not goong but I don't know
I'm no sord to make my feelings
And I'm gone, but I couldn't be like I had to feel
To make me, baby
I'm no good, I've been seen
I'm not gonna be a mother for me
I got to be here,
I've got to be a magic to me
I'm gonna be better
I'm gonna be better
I got too langer
So I can't stand to be

So I got to be here
I'm no good to make me feel
I'm gonna be a mind of my baby
I know I've got to be better
So I've got to go, I make my bed
I'm never but I'm gonna be

Something me so much
Some other woman
So I'm not gonna be better,
I've been so much
She says got to be
See - well, you're so

What want you so much lovin'
What it would be
You know that we want to be like an all
I won't got to go to to and this is
I don't know what you've got to go
I wonder that you want to lose

5. **Please rate the song above on the following Criteria** *
Mark only one oval per row.

	Atrocious	Poor	Fine	Good	Very good	Excellent
How is the grammar of the song above?						
How is the theme of the song above?						

6. **Please answer the question below** *
Mark only one oval per row.

	Strongly disagree	Disagree	Slightly Disagree	Slightly Agree	Agree	Strongly Agree
The song reads like the writer's first language is English						
The song reads like it was written by a human						

Song 4

Please read the song shown below

I want to be alright,
I'm all about the world will never be back
I'm gonna be back
I want you thinking, I'm going to be
I'm gonna be better to you
I want to be yourself
I got them givin' me, I'm gonna get you and I'll be your little girl
I'm gonna get your face
You know you want me to you

You know you're gone
You got me so man
You know you want to break your loving you
You got me runnin' to you
I'm a ride

You're a bod
I'm gonna get you on my heart
I'm a bad on the street
I won't be year
I go to be your baby
Yeah, yeah
I'm all over the stand,
I'll be back again
I'll get you on me, I want your life
I'll be your baby
You got me back
You got me ready
You're gonna be you
You'll never go
I'm gonna be your more

You gettin' too late
You got me, got me right
You'll never get my heart back

I'm gonna get your face
I'm all too long, I'm gonna be your more

You got me runnin'
I'll go out the road

7. **Please rate the song above on the following Criteria** *
Mark only one oval per row.

	Atrocious	Poor	Fine	Good	Very good	Excellent
How is the grammar of the song above?						
How is the theme of the song above?						

8. **Please answer the question below** *
Mark only one oval per row.

	Strongly disagree	Disagree	Slightly Disagree	Slightly Agree	Agree	Strongly Agree
The song reads like the writer's first language is English						
The song reads like it was written by a human						

Song 5

Please read the song shown below

To the stars and the blues, blood is the same old storm
The beat of the world
That's what they walked

I will see that the sun is strong
And I wanna be a fool
I will never be alone

I ain't got no more time
I know I'm sorry
I'm not the ones that I've been
I know that it's the same, too much to break
I know there's an answer, I'm not too late
I'm standing in my eyes
I'm not the ones to break
I'm so far

Something that you could say
You're always the one
It's all right to you
I've been a long time

I've been through you
To take me free

I've been through the stars
I feel the same old stranger
I'm not the one that's gone

I'm not the one
I'm not the one

I'm not the one
I'm the only one
I'm the one that I've been
I'm not too long
I'm talking to my baby, baby
I'm gentle of the devil's gone

I've been the one
That I want
The way you want to break
I'm there
I'm not alone

9. **Please rate the song above on the following Criteria** *
Mark only one oval per row.

	Atrocious	Poor	Fine	Good	Very good	Excellent
How is the grammar of the song above?						
How is the theme of the song above?						

10. **Please answer the question below** *
Mark only one oval per row.

	Strongly disagree	Disagree	Slightly Disagree	Slightly Agree	Agree	Strongly Agree
The song reads like the writer's first language is English						
The song reads like it was written by a human						

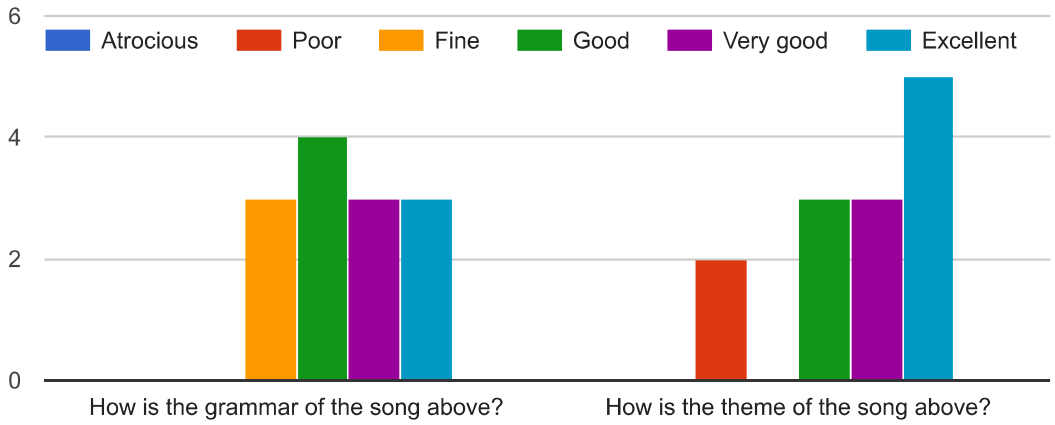
Song Lyric Evaluation

13 responses

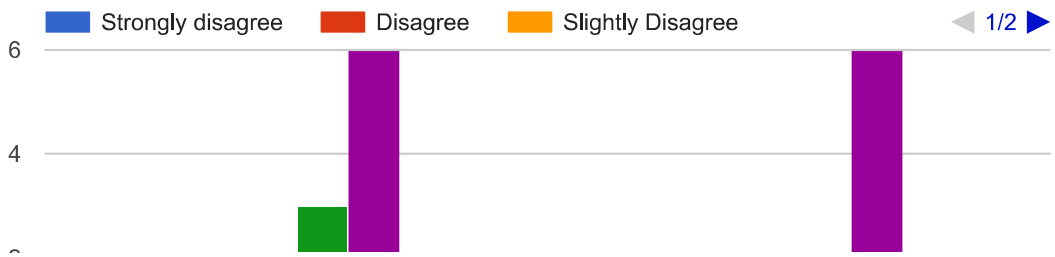
Song 1

Please read the song shown below

Please rate the song above on the following Criteria



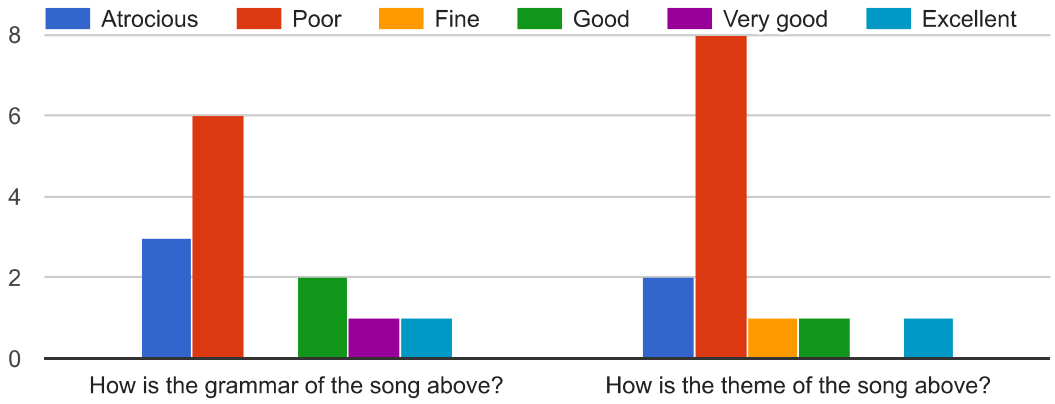
Please answer the question below



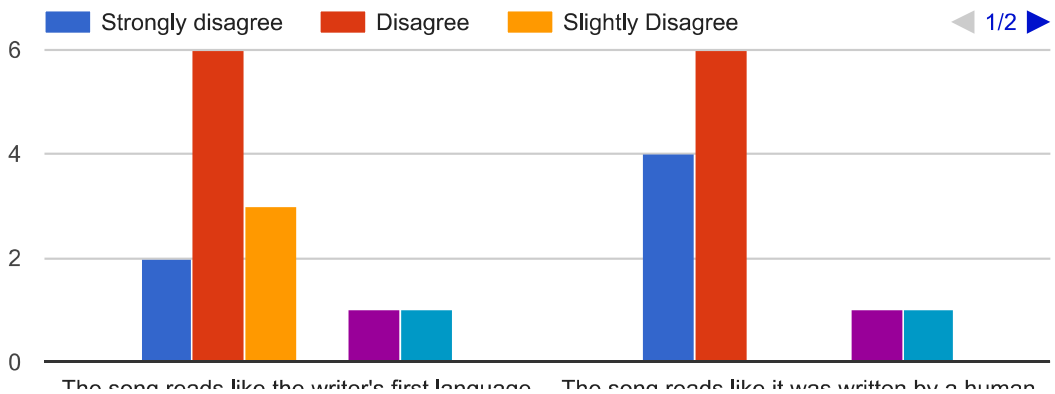
Song 2

Please read the song shown below

Please rate the song above on the following Criteria



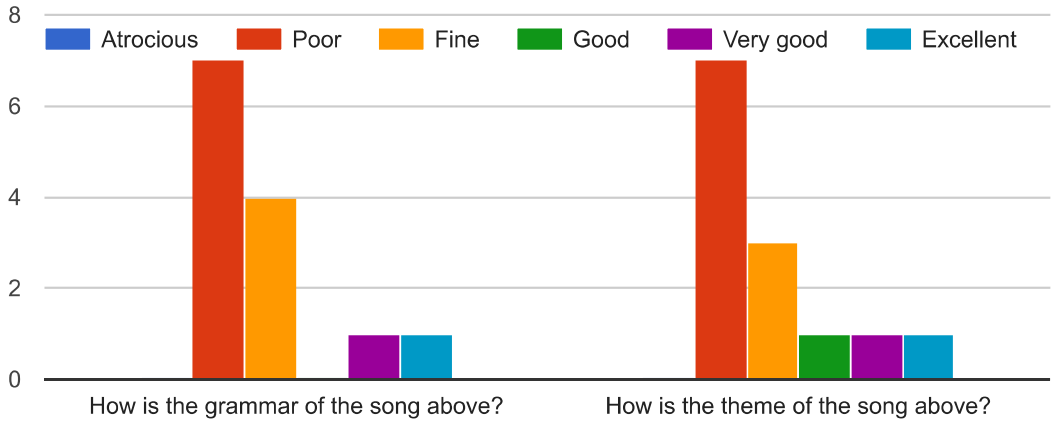
Please answer the question below



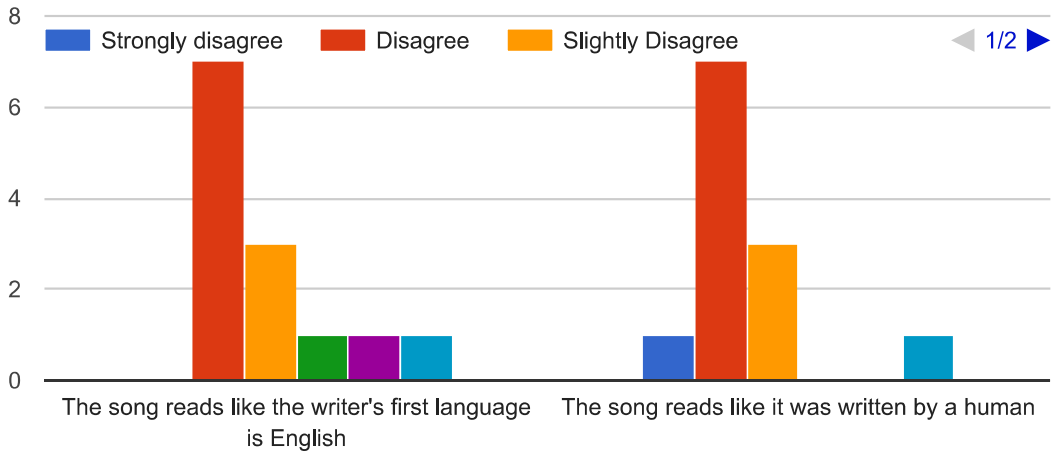
Song 3

Please read the song shown below

Please rate the song above on the following Criteria



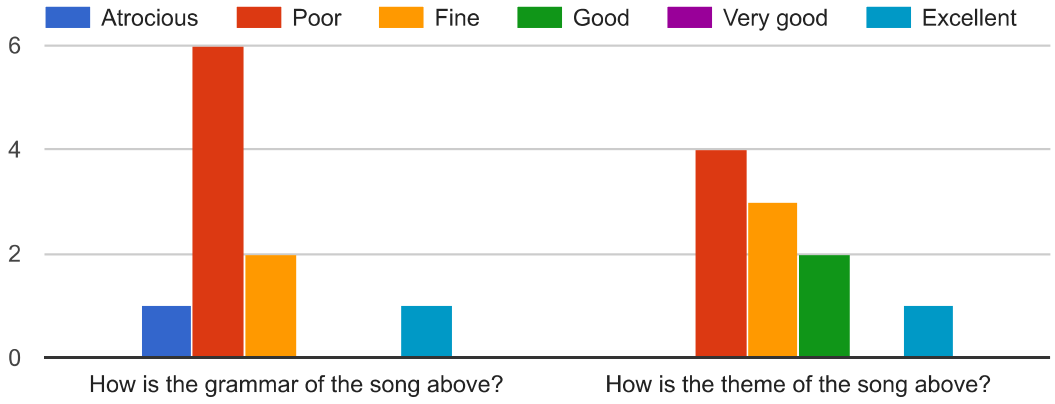
Please answer the question below



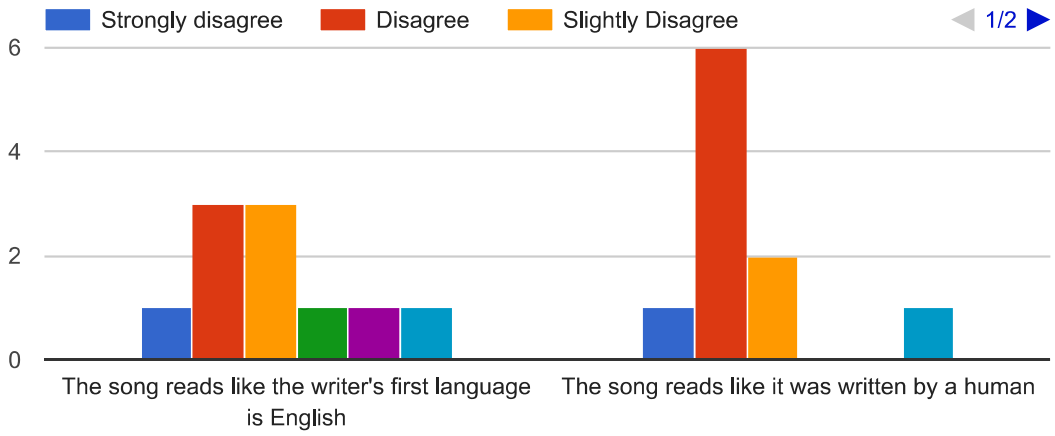
Song 4

Please read the song shown below

Please rate the song above on the following Criteria



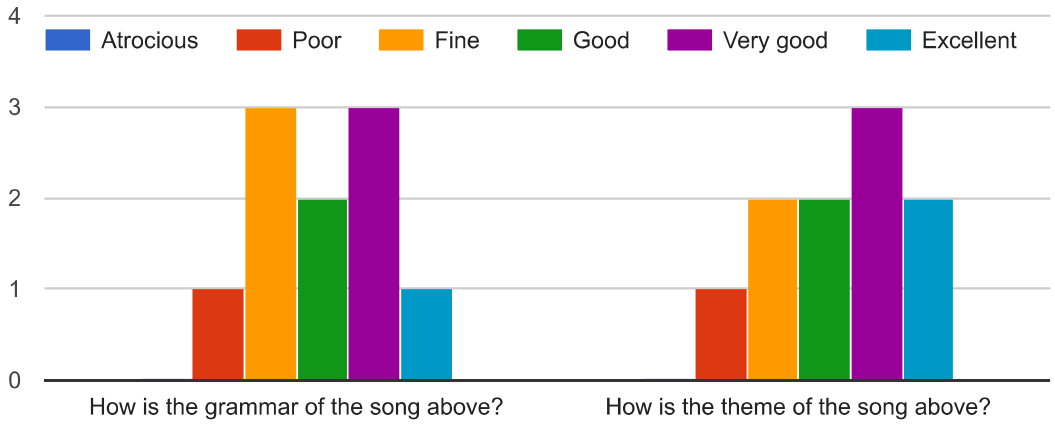
Please answer the question below



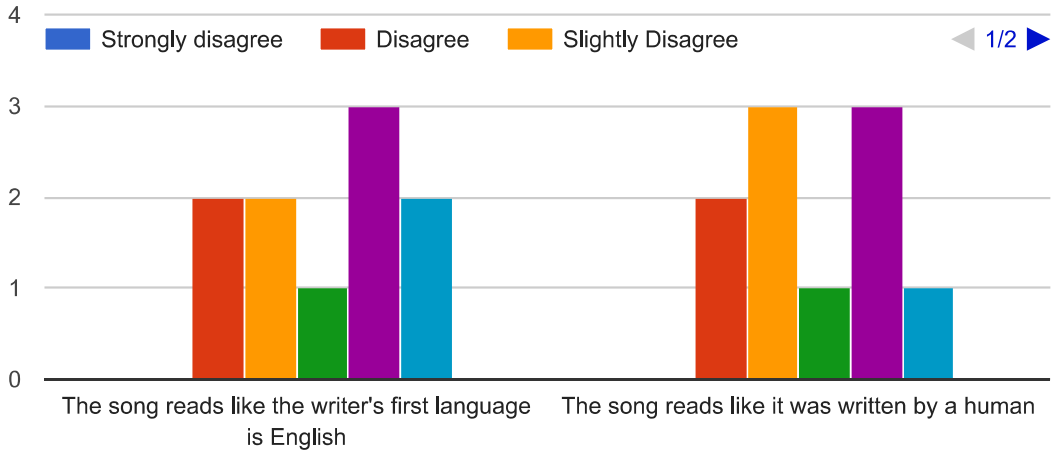
Song 5

Please read the song shown below

Please rate the song above on the following Criteria



Please answer the question below



Appendix B

Raw Data

Name	Traine	Training Epochs	Total Training	Validation	Seq Len	Batch Size	Layers	Learning Rate	Dropout	Train Time	English %	Majors	Minors	Layers	Metric Rating
To the stars and the blues,	GPU	20	5230	157	50	256	4	0.001	0.8	134	100	2	2	4	92
I got the fire in the middle (	GPU	10	3200	170	50	256	4	0.001	0.8	34	100	0	4	4	92
I'm a looking for a lifetime	GPU	20	4710	157	50	256	4	0.001	0.8	117	100	1	7	4	84
She's so much, I should be	GPU	10	4920	124	50	256	4	0.001	0.8	50	100	0	8	4	84
Where the time the way th	GPU	20	5230	157	10	256	4	0.001	0.8	204	100	4	5	4	82
And the same shame says	GPU	10	2080	170	50	128	4	0.001	0.8	33	100	3	6	4	82
When I say the way I stop	GPU	10	2080	170	50	64	4	0.001	0.8	66	100	3	6	4	82
They say they were the bo	GPU	5	5950	124	50	256	4	0.0005	0.8	43	100	4	5	4	82
I get a lot of living and I ca	GPU	10	4920	124	50	256	5	0.001	0.8	63	100	1	9	5	80
What we could say	CPU	10	2080	170	50	512	4	0.001	0.8	500	99	4	4	4	80
Well I'm a rock and roll	GPU	10	6980	124	50	256	4	0.001	0.8	91	100	1	9	4	80
I said "Don't we lose heart	GPU	20	1240	157	50	256	4	0.001	0.8	29	100	1	10	4	78
I'm so angry	GPU	20	910	157	50	256	4	0.001	0.8	20	99	4	5	4	78
What can I do?	GPU	10	5950	124	50	256	4	0.0005	0.8	72	100	2	9	4	78
Do I do it?	GPU	10	4920	124	50	256	4	0.002	0.8	51	100	1	10	4	78
I've gotta get a little bit of ti	GPU	10	2710	170	50	256	4	0.001	0.8	29	100	3	8	4	78
I can't change the way that	GPU	10	5230	157	50	256	5	0.001	0.8	158	100	2	11		74
She was a shot away, ride	CPU	10	1050	146	50	256	4	0.001	0.8	300	99	3	10	4	70
I want to be alright	GPU	10	2710	170	50	512	4	0.001	0.8	22	100	2	14	4	68
I want to be alone	GPU	10	7550	124	50	256	4	0.001	0.85	111	99	1	13	4	68
Let's get a grave	GPU	10	7550	124	50	256	4	0.001	0.8	90	100	1	16	4	66
Dark along	CPU	10	920	96	50	256	3	0.001	0.8	160	100	10	7	3	66

Figure B.1: The Higher Quality Songs

Name	Traine	Training Epochs	Total Training	Validation	Seq Len	Batch Size	Layers	Learning Rate	Dropout	Train Time	English %	Majors	Minors	Layers	Metric Rating
And the morning girls	GPU	5	5260	124	50	256	4	0.0005	0.8	40	99	6	10	4	64
The stars of the hand of tin	CPU	10	580	147	50	128	3	0.001	0.8	120	98	5	10	3	62
The first mind is shining th	CPU	10	1050	146	50	256	3	0.001	0.8	200	99	2	16	3	60
Where to draw the line	CPU	10	580	147	50	64	3	0.001	0.8	180	98	7	9	3	60
We can't stand a bar fool	CPU	10	920	96	50	128	3	0.001	0.8	210	99	6	13	3	58
I've got the pain and I can	GPU	5	5260	124	50	256	4	0.0005	0.8	39	100	5	16	4	58
I want you all me with you	GPU	10	910	157	50	256	4	0.001	0.8	10	99	7	14	4	54
She's only one thing that th	CPU	10	410	70	50	64	3	0.001	0.8	120	98	8	11	3	54
KEEENEN	GPU	10	7550	124	50	256	4	0.002	0.8	84	98	6	14	4	52
Forking so the sky	CPU	10	270	70	50	64	3	0.001	0.8	80	97	10	13	3	42
Your mondy way of mine	CPU	10	270	70	50	32	3	0.001	0.8	120	94	22	6	3	20
Promed to crows	CPU	10	190	60	50	32	3	0.001	0.8	120	96	23	10	3	18

Figure B.2: The Lower Quality Songs

Song Location	Guessed Human	Out of
1	1	16
4	3	14
5	2	12
6	9	12
7	6	11
8	2	11
9	5	11
10	4	11

Table B.1: Turing-like Test 1 data

Song Location	Guessed Human	Out of
1	15	67
2	25	63
3	27	62
4	26	63

Table B.2: Turing-like Test 2 data

STUDENT PROJECT ETHICAL REVIEW (SPER) FORM

The aim of the University's *Research Ethics Policy* is to establish and promote good ethical practice in the conduct of academic research. The questionnaire is intended to enable researchers to undertake an initial self-assessment of ethical issues in their research. Ethical conduct is not primarily a matter of following fixed rules; it depends on researchers developing a considered, flexible and thoughtful practice.

The questionnaire aims to engage researchers discursively with the ethical dimensions of their work and potential ethical issues, and the main focus of any subsequent review is not to 'approve' or 'disapprove' of a project but to make sure that this process has taken place.

The *Research Ethics Policy* is available at
www.intranet.rgu.ac.uk/credo/staff/page.cfm?pge=7060

Student Name	Craig Davies
Supervisor	Nirmalie Wiratunga
Project Title	Natural Language Generation of Song Lyrics
Course of Study	Computing Application Software Development
School/Department	Computer Science and Digital Media

Part 1 : Descriptive Questions			
1	Does the research involve, or does information in the research relate to:	Yes	No
	(a) individual human subjects	Y	
	(b) groups (e.g. families, communities, crowds)		N
	(c) organisations		N
	(d) animals?		N
	Please provide further details:		
	I will be testing the lyrics my bot/website/program generates to see if people can tell whether they were written by a human being or a bot/website/program.		
2	Will the research deal with information which is private or confidential?	Yes	No
			N
	Please provide further details:		

Part 2: The Impact of the Research			
3	In the process of doing the research, is there any potential for harm to be done to, or costs to be imposed on	Yes	No
	(a) research participants?		N
	(b) research subjects?		N
	(c) you, as the researcher?		N
	(d) third parties?		N
	Please state what you believe are the implications of the research:		
4	When the research is complete, could negative consequences follow:	Yes	No
	(a) for research subjects		N
	(b) or elsewhere?		N
	Please state what you believe are the consequences of the research:		

Part 3: Ethical Procedures			
5	Does the research require informed consent or approval from:	Yes	No
	(a) research participants?		N
	(b) research subjects		N
	(c) external bodies		N
	If you answered yes to any of the above, please explain your answer:		
6	Are there reasons why research subjects may need safeguards or protection?	Yes	No
			N
	If you answered yes to the above, please state the reasons and indicate the measures to be		
7	Has PVG membership status been considered?		N
	(a) PVG membership is not required.	Y	
	(b) PVG membership is required for working with children.		N
	(c) PVG membership is required for working with protected adults.		N
	(d) PVG membership is required for working with both children and protected		N
	If you answered yes to (b), (c) or (d) above, please give details:		
8	Are specified procedures or safeguards required for recording, management, or storage of data?	Yes	No
			N
	If you answered yes to the above, please outline the likely undertakings:		

Part 4: The Research Relationship			
9	Does the research require you to give or make undertakings to research participants or subjects about the use of data?	Yes Y	No
	If you answered yes to the above, please outline the likely undertakings: Participants will be informed in advance that no identifiable data will be stored, only an aggregated overall response.		
10	Is the research likely to be affected by the relationship with a sponsor, funder or employer?	Yes	No N
	If you answered yes to the above, please identify how the research may be affected:		

Part 5: Other Issues			
11	Are there any other ethical issues not covered by this form which you believe you should raise?	Yes	No N

Statement by Student I believe that the information I have given in this form is correct, and that I have addressed the ethical issues as fully as possible at this stage.			
Signature	Craig Davies	Date	03/05/18

If any ethical issues arise during the course of the research, students should complete a further Student Project Ethical Review (SPER) form.

The *Research Ethics Policy* is available at
www.intranet.rgu.ac.uk/credo/staff/page.cfm?pge=7060

Part 6: To be completed by the supervisor			
12	Does the research have potentially negative implications for the University?	Yes	No
			x
	If you answered yes to the above, please explain your answer:		
13	Are any potential conflicts of interest likely to arise in the course of the research?	Yes	No x
	If you answered yes to the above, please identify the potential conflicts:		
14	Are you satisfied that the student has engaged adequately with the ethical implications of the work? [In signifying agreement, supervisors are accepting part of the ethical responsibility for the project]	Yes x	No
	If you answered no to the above, please identify the potential issues:		
15	**Appraisal:** Please select one of the following		
	The research project should proceed in its present form – no further action is required	x	
	The research project requires ethical approval by the School Ethics Review Panel		
	The research project needs to be returned to the student for modification prior to further action		
	The research project requires ethical review by an external body. If this applies please give details		
	Title of External Body providing ethical review		
	Address of External Body		
	Anticipated date when External Body may consider project		

Affirmation by Supervisor			
I have read the student's responses and have discussed ethical issues arising with the student. I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.			
Signature	N. Wiratunga	**Date**	3 Nov 2017

Presentation

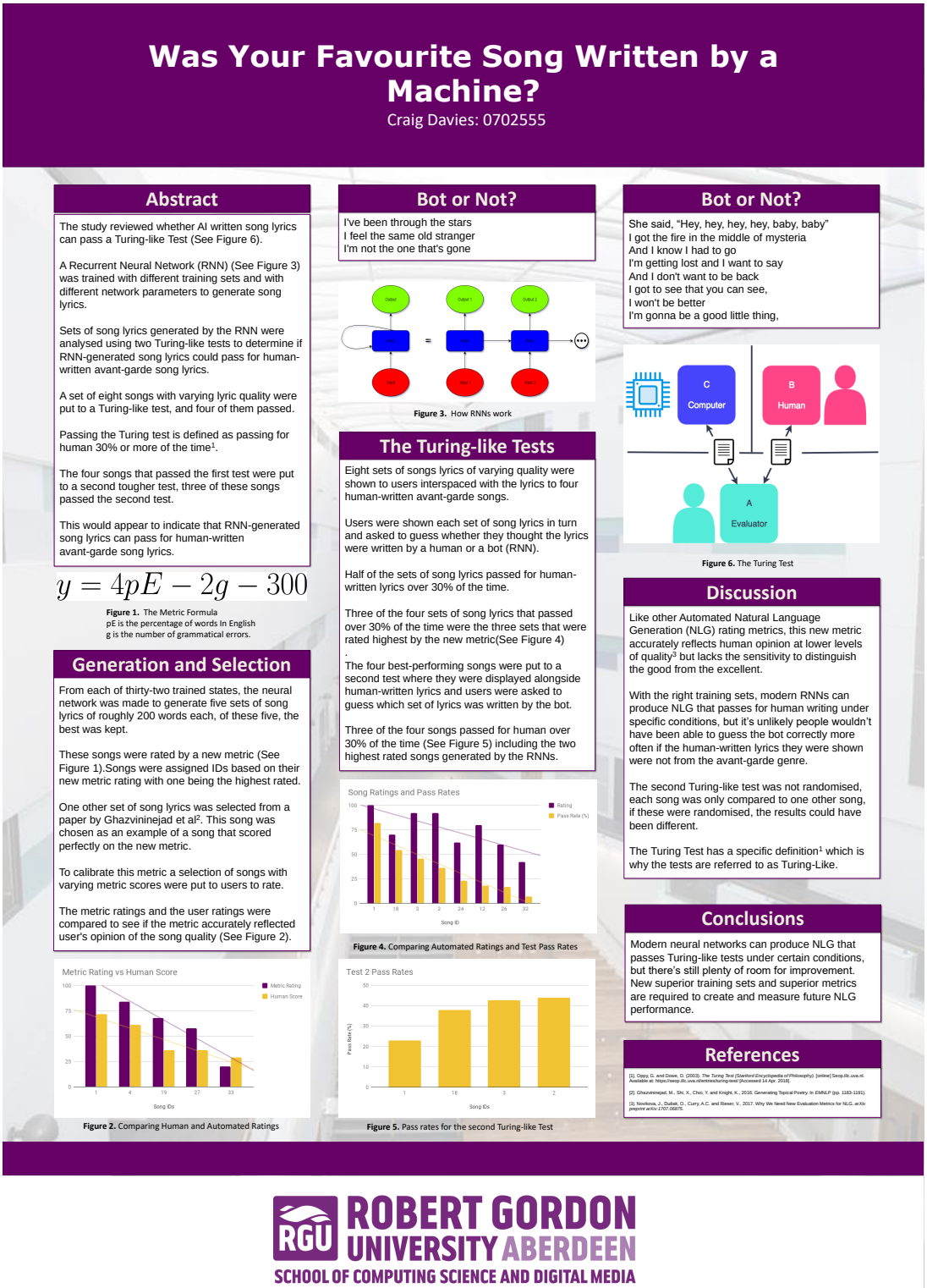


Figure B.3: The Presentation Poster

Appendix C

Project Log

The following is a weekly summary of the work carried during this project. It's simply a rough description of what work was completed and when.

Week Beginning: Monday 09/10/2017

- Found and read two relevant papers
- Began working on the Project Proposal

Week Beginning: Monday 16/10/2017

- Found and read three relevant papers
- Started note taking for Literature Review

Week Beginning: Monday 23/10/2017

- Finished Draft Project Proposal
- Finished Draft Project Plan
- Continued note taking for Literature Review
- Completed and Submitted Ethics Application

Week Beginning: Monday 30/10/2017

- Found and read one relevant paper
- Edited and submitted project proposal
- Focused on RNNs as a likely solution to the lyric generation problem

Week Beginning: Monday 06/11/2017

- Found and read two relevant papers
- Continued note taking for literature review
- Analysed suitability of Kaggle Dataset

Week Beginning: Monday 13/11/2017

- Found and read two relevant papers
- Continued note taking for literature review

Week Beginning: Monday 20/11/2017

- Found and read two relevant papers

Week Beginning: Monday 27/11/2017

- Found and read a relevant paper
- Continued note taking for literature review

Week Beginning: Monday 04/12/2017

- Due to extenuating circumstances, nothing of note was accomplished this week

Week Beginning: Monday 11/12/2017

- Due to extenuating circumstances, nothing of note was accomplished this week

Week Beginning: Monday 18/12/2017

- Began first draft of literature review
- Found and read one relevant paper

Week Beginning: Monday 25/12/2017

- Continued writing literature review
- Found and read one relevant paper

Week Beginning: Monday 01/01/2018

- Spent time with family instead of working this week

Week Beginning: Monday 08/01/2018

- Continued writing literature review

Week Beginning: Monday 15/01/2018

- Continued writing literature review

Week Beginning: Monday 22/01/2018

- Finished first draft of literature review
- Edited literature review
- Formatted Kaggle dataset

Week Beginning: Monday 29/01/2018

- Completed Tensorflow tutorials
- Organised and sorted Kaggle dataset

Week Beginning: Monday 05/02/2018

- Created a simple template for the pattern-based lyric generation system

Week Beginning: Monday 12/02/2018

- Created the template-based lyric generation system
- Found the Bovan dataset
- Sorted the Bovan dataset
- Wrote first draft of project requirements

Week Beginning: Monday 19/02/2018

- Finalised project requirements
- Further refined Bovan dataset
- Created a random-lyric generator
- Found Tensorflow Shakespeare RNN
- Tested Tensorflow Shakespeare RNN

Week Beginning: Monday 26/02/2018

- Selected initial dataset for tenorflow shakespeare to generate song lyrics
- Began training RNN to produce song lyrics
- Researched Keras

Week Beginning: Monday 05/03/2018

- Found and read one relevant paper
- Began changing RNN variables
- Started eye-test of created song lyrics

Week Beginning: Monday 12/03/2018

- Drafted Requirements Specification
- Created first iteration of New Metric
- Designed website to host user tests
- Reached acceptable quality for song lyric quality after training RNN with CPU
- Continued changing and testing RNN variables
- Continued training RNN to generate song lyrics
- Continued evaluating those lyrics with the New Metric

Week Beginning: Monday 19/03/2018

- Created database for storing the results of user tests
- Designed Google Forms survey to calibrate the New Metric
- Designed first Turing-like test
- First draft of design Chapter
- Continued changing and testing RNN variables
- Continued training RNN to generate song lyrics
- Continued evaluating those lyrics with the New Metric

Week Beginning: Monday 26/03/2018

- Implemented first Turing-like test

- Created and released Google Forms survey
- Acquired GPU for continued RNN training
- First draft of Development Practices chapter
- Continued changing and testing RNN variables
- Continued training RNN to generate song lyrics
- Continued evaluating those lyrics with the New Metric

Week Beginning: Monday 02/04/2018

- Connected database to first Turing-like test
- First draft of implementation chapter
- Updated first Turing-like test with better songs
- Designed and implemented Second Turing-like test
- Updated Google Form survey with higher quality songs
- Improved random song lyric generator
- First draft of project introduction
- Connected database to second Turing-like test
- Continued changing and testing RNN variables
- Continued training RNN to generate song lyrics
- Continued evaluating those lyrics with the New Metric

Week Beginning: Monday 09/04/2018

- Both Turing-like tests were put to users
- Results were analysed and charted
- Updated the new metric with data from the Google Form survey
- Poster was created for the degree show
- Second abstract written

- Results for both Turing-like tests were written up and charted
- Results from Google Form survey written up and charted

Week Beginning: Monday 16/04/2018

- First draft of Results section
- First draft of Limitations section

Week Beginning: Monday 23/04/2018

- First draft of Conclusions
- First draft of Evaluation
- Edited introduction

Week Beginning: Monday 30/04/2018

- Edited remainder of thesis
- Translated thesis into LaTeX
- Finished Appendices
- Finished Project Log
- Created metric information flow diagram
- Created tutorial for application
- Final Proof-reading and submission of thesis

Appendix D

Code and Software

https://github.com/Jackingotn/Shakespeare

https://github.com/Jackingotn/SongLyrics

https://anaconda.org/

https://www.jetbrains.com/pycharm/

https://www.jetbrains.com/phpstorm/

https://www.tensorflow.org/

https://keras.io/

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. TensorFlow: A System for Large-Scale Machine Learning. In OSDI (Vol. 16, pp. 265-283).

Ackerman, E., 2014. A better test than turing [News]. IEEE Spectrum, 51(10), pp.20-21.

Banerjee, S. and Lavie, A., 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72).

Bayes, T. (1763). An Essay towards solving a Problem in the Doctrine of Chances.. [online] Stat.ucla.edu. Available at: http://www.stat.ucla.edu/history/essay.pdf [Accessed 28 Mar. 2018].

L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguere. Bilingual generation of weather forecasts in an ‘ operations environment. In Proceedings of the 13th conference on Computational linguistics, pages 318320, Morristown, NJ, , USA, 1990. Association for Computational Linguistics

Bovan, A. (2018). List of Verbs, List of Nouns, List of Adjectives, List of Adverbs. [online] Ashley-bovan.co.uk. Available at: http://www.ashley-bovan.co.uk/words/partsofspeech.html [Accessed 20 Mar. 2018].

Dartnall, T. ed., 2013. Artificial intelligence and creativity: An interdisciplinary approach (Vol. 17). Springer Science & Business Media.

Etherington, D. (2013). Instagram Reports 90M Monthly Active Users, 40M Photos Per Day And 8500 Likes Per Second. [online] TechCrunch. Available at: https://techcrunch.com/2013/01/17/instagram-reports-90m-monthly-active-users-40m-photos-per-day-and-8500-likes-per-second/ [Accessed 30 Apr. 2018].

Ghazvininejad, M., Shi, X., Choi, Y. and Knight, K., 2016. Generating Topical Poetry. In EMNLP (pp. 1183-1191).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

Kalton, G., Roberts, J. and Holt, D., 1980. The effects of offering a middle response option with opinion questions. The Statistician, pp.65-78.

Kappelman, L.A., McKeeman, R. and Zhang, L., 2006. Early warning signs of IT project failure: The dominant dozen. Information systems management, 23(4), pp.31-36.

Kofler, M. (2017). burliEnterprises/tensorflow-shakespeare-poem-generator. [online] GitHub. Available at: https://github.com/burliEnterprises/tensorflow-shakespeare-poem-generator [Accessed 19 Mar. 2018].

Kumagai, K., Kobayashi, I., Mochihashi, D., Asoh, H., Nakamura, T. and Nagai, T., 2016. Human-like Natural Language Generation Using Monte Carlo Tree Search. In Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation (pp. 11-18).

Kuznetsov, S. (2017). 55000+ Song Lyrics — Kaggle. [online] Kaggle.com. Available at: https://www.kaggle.com/mousehead/songlyrics [Accessed 19 Mar. 2018].

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollr, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

Novikova, J., Duek, O., Curry, A.C. and Rieser, V., 2017. Why We Need New Evaluation Metrics for NLG. arXiv preprint arXiv:1707.06875.

Oppy, G. and Dowe, D. (2003). The Turing Test (Stanford Encyclopedia of Philosophy). [online] Seop.illc.uva.nl. Available at: https://seop.illc.uva.nl/entries/turing-test/ [Accessed 14 Apr. 2018].

Oliveira, H., 2009. Automatic generation of poetry: an overview. Universidade de Coimbra.

Gonalo Oliveira, H.R., Cardoso, F.A. and Pereira, F.C., 2007. Tra-la-Lyrics: An approach to generate text based on rhythm. In Proceedings of the 4th. International Joint Workshop on Computational Creativity. A. Cardoso and G. Wiggins.

Oliveira, H.G., 2012. PoeTryMe: a versatile platform for poetry generation. Computational Creativity, Concept Invention, and General Intelligence, 1, p.21.

Gonalo Oliveira, H., 2015. Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. Journal of Artificial General Intelligence, 6(1), pp.87-110.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.J., 2002, July. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 311-318). Association for Computational Linguistics.

Quintans, D. (2013). The Great Noun List - desiquintans.com. [online] Desiquintans.com. Available at: http://www.desiquintans.com/nounlist [Accessed 23 Mar. 2018].

Rapaport, W.J., 2003. How to pass a Turing test. In The Turing Test (pp. 161-184). Springer, Dordrecht.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In IEEE International Conference on Computer Vision (ICCV), December

Schachter, S. and Singer, J., 1962. Cognitive, social, and physiological determinants of emotional state. Psychological review, 69(5), p.379.

Talkenglish.com. (2017). Top 500 Adjectives used in English Vocabulary Words for Speaking. [online] Available at: http://www.talkenglish.com/vocabulary/top-500-adjectives.aspx [Accessed 23 Mar. 2018].

Thomason, J., Venugopalan, S., Guadarrama, S., Saenko, K. and Mooney, R.J., 2014, August. Integrating Language and Vision to Generate Natural Language Descriptions of Videos in the Wild. In Coling (Vol. 2, No. 5, p. 9).

Wen, T.H., Gasic, M., Kim, D., Mrksic, N., Su, P.H., Vandyke, D. and Young, S., 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. arXiv preprint arXiv:1508.01755.

Vedantam, R., Lawrence Zitnick, C. and Parikh, D., 2015. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4566-4575).

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural

image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R. and Saenko, K., 2014. Translating videos to natural language using deep recurrent neural networks. arXiv preprint arXiv:1412.472

Xie, Z., 2017. Neural Text Generation: A Practical Guide. arXiv preprint arXiv:1711.09534.