

gpt2-japanese と Word2Vec を用いた界限曲の 作詞システムの構築

学籍番号：1 F10190019

氏名：泉川 叶多

指導教員名：カン フェルドウス

論文要旨

ネット上にはアーティストである全てあなたの所為です。の曲を模倣する文化があり今日、2000 曲以上の模倣した曲が存在する。しかし模倣する上での問題があり、それはその曲の歌詞のユニークな特徴を理解する必要がある為、作詞をすることが難しいことだ。そこで本研究では「カテゴリタイプ」、「ソングタイプ」、「モデルタイプ」の計 3 つの生成タイプの歌詞生成システムを構築した。そして特徴をよく理解できていない人でも簡単にその歌詞生成システムを利用できるウェブアプリケーション「全て歌詞の所為です。」を制作し公開した。カテゴリタイプとソングタイプには単語ベクトルを扱うことができる言語モデル Word2Vec と自作した歌詞生成アルゴリズムを用いて、全てあなたの所為です。の曲の歌詞に似た歌詞を生成する。両者の違いは歌詞生成に使う単語の選び方にある。ソングタイプは全てあなたの所為です。のうちユーザーが指定する 1 曲と、その曲を模倣している曲の歌詞のうち、その歌詞に登場する単語自体とその単語を連想させる単語を歌詞生成に利用する。一方、ソングタイプは全てあなたの所為です。やその模倣曲を含めた界限曲全体で模倣関係を全てのノードの重みが 1 の無向グラフで表し、そのグラフを任意の条件下で探索することで曲を選択し、それらの曲に登場する単語とその単語を連想させる単語を歌詞生成に使う。他方、モデルタイプではファインチューニングを行うことができる gpt2-japanese 言語モデルを用いて歌詞を生成するシステムを構築する。評価に関しては主観評価と客観評価の 2 つ評価を行った。主観評価では全て歌詞の所為です。上で生成した歌詞を事前に用意したメロディにつけ、使い勝手の良し悪しを調査する。客観評価では、生成した歌詞を全て歌詞の所為です。上で 1 行ごと評価することができようにし、コミュニティーに所属している方々を中心に評価をしてもらった。結果、一番評価が高かった生成タイプは言語モデル gpt2-japanese を用いたモデルタイプであった。また、カテゴリタイプとソングタイプの中でも原曲が支離滅裂な歌詞は評価が高くなり、逆にストーリー性がある歌詞の原曲のカテゴリは一部例外を除いて評価が低くなる傾向があった。

1.	はじめに	6
1.1.	背景と問題.....	6
1.2.	研究の目的.....	7
1.3.	貢献.....	7
1.4.	本論文の構成.....	8
2.	関連研究	8
2.1.	ユーザーが条件を指定可能な歌詞の候補を提示するソフトウェアに関する研究.....	8
2.2.	事前学習済み言語モデルを用いて歌詞を生成する研究	8
2.3.	SeqGAN を用いた歌詞生成の研究.....	9
2.4.	n-gram 言語モデルに基づいた歌詞生成の研究	9
2.5.	パロディ音楽の歌詞を生成する研究.....	9
3.	設計と実装.....	9
3.1.	曲のデータの登録	10
3.2.	模倣単語の生成	11
3.2.1.	歌詞の整形.....	12
3.2.2.	歌詞の分かち書き	12
3.2.3.	各単語の品詞分析.....	12
3.2.4.	単語のフィルタリング	13
3.2.5.	模倣単語の末尾の編集	13
3.2.6.	模倣単語の探索.....	13
3.2.7.	模倣単語の保存.....	14
3.2.8.	全て歌詞の所為です。への POST.....	14
3.3.	歌詞のファインチューニング [12]	15
3.3.1.	セットアップの実行	15
3.3.2.	歌詞のダウンロードと整形	15
3.3.3.	データセットの作成	16

3.3.4.	ファインチューニングの実行	16
3.4.	歌詞生成の実装	16
3.5.	全て歌詞の所為です。の実装	27
3.5.1.	ページについて	27
3.5.1.1.	トップページ	27
3.5.1.2.	曲の一覧と検索ページ	29
3.5.1.3.	曲ページ	29
3.5.1.4.	歌詞の登録ページ	30
3.5.1.5.	曲データの追記ページ	31
3.5.1.6.	歌詞データの修正と削除提案ページ	32
3.5.1.7.	チャンネルページ	33
3.5.1.8.	歌詞の生成ページ	34
3.5.1.9.	生成結果ページ	35
3.5.1.10.	生成された歌詞ページ	36
3.5.2.	データベース設計	37
3.6.	研究全体のシステム図	37
4.	評価	38
4.1.	主観評価	38
4.1.1.	主観評価の評価方法	38
4.1.2.	作曲した結果	39
4.1.3.	全て歌詞の所為です。を使ってみての評価	40
4.2.	客観評価	40
4.2.1.	客観評価の評価方法	40
4.2.2.	評価結果と考察	41
4.2.2.1.	カテゴリタイプの評価	41
4.2.2.2.	ソングタイプの評価	43
4.2.2.3.	カテゴリタイプとモデルタイプの評価について	44
4.2.2.4.	モデルタイプの評価	44
5.	結論	45
5.1.	まとめ	45

5.2. 今後の課題.....	46
-----------------	----

1. はじめに

1.1. 背景と問題

ここ二十数年のネットの普及により様々なコミュニティが発生した。その一つである「界限曲」とは全てあなたの所為です。を中心としたアーティストの作風を模倣するジャンル、及びそのコミュニティである。現在、コミュニティの中心部である「Imitate Community」の Discord サーバーには 268 人が所属しており、少なくとも 170 人以上のアーティスト（以下、界限の人と称す）が模倣した曲（以下、模倣曲と称す）を 2000 曲以上リリースしており今日、コミュニティは発展し続けている。

全てあなたの所為です。の曲（以下、原曲と称す）、及び模倣曲の歌詞には他のアーティストには無い特徴がある。

1 つ目は歌詞が意味不明なところだ(以下、特徴 1 と称す)。原曲の歌詞は非文ではない、つまり文法的な間違いの無い文だが、歌詞全体に脈絡が無く支離滅裂なことが特徴だ。支離滅裂の度合いは各曲に違うが例えば『エヌ』の場合は、

「因数分解とオタマジャクシは、刃物と日記を混ぜました。」 [1]

といった通常の文章には登場しないような単語の連なり方をしている。しかし曲によっては支離滅裂ではあるが一部、ストーリー性がある歌詞を含む曲も存在し、例えば『K²』の場合は

「形など無いはずの、憾がどろりと。こちらを向き、見つめていた。幽き声の詩が、幾重にも重なり。一つの意味になるのでしょうか。」 [2]

のように各文に注目すると若干のストーリー性があることが分かる。

2 つ目は歌詞として使われている単語が難解なことだ(以下、特徴 2 と称す)。例えば合目的的・ラザロ徴候・第四の壁・門前雀羅を張るといった日本語がネイティブな人でも知らないマニアックな単語が歌詞に使われることが多い。

3 つ目は歌詞に使われている単語は漢字をひらかず、漢字表記にすることだ(以下、特徴 3 と称す)。例えば所為（せい）・只管（ひたすら）・徐（おもむろ）といったように、一般的にはひらがな表記にする単語を漢字に直すことが多い。

4 つ目は模倣曲の歌詞に使う単語に原曲の歌詞で使われる単語そのもの、もしくは類語といったその単語を連想させる単語(以下、これらの単語を模倣単語と称す)を使うことだ(以下、特徴 4 と称す)。比較の一例として原曲の一つである『...』と

その模倣曲である『|||』を挙げるとすると、歌詞の冒頭はそれぞれ

「穴の空いた両の手で、喉の渇きは潤せず、甘いはずの水は、掬っても零れてゆく。」

[3]

「手のひらに穴が開いて、杜撰な真実に気がついた。視界に見えるものだけが、にやりと嘲笑って見えた。」 [4]

となっている。この2つの曲は歌詞の共通点は両方とも「手」と「穴」が登場することだ。つまり『|||』は歌詞の冒頭でこれら2つの単語を原曲から借用しているといえる。

最後に5つ目は原曲以外にも模倣曲の歌詞も模倣されることだ(以下、特徴5と称す)。模倣曲は原曲の二次創作といえるがそれだけではなく模倣曲をさらに模倣した三次創作、さらに三次創作を模倣した四次創作、というように創作が再帰的に行われる特徴があり、模倣されたとき歌詞も模倣される。

さて、模倣曲の作詞において以上で述べたような5つの特徴を持った歌詞を作詞する必要があるが、これは原曲やその模倣曲に詳しくないと作詞することは困難だ。模倣曲全体として各特徴をどのように、またどのタイミング(Aメロ・Bメロ・サビ等)で継承するかは一種の定石が存在し、それを理解する為には原曲とその界限曲を数十曲聴かなければならないだろう。

1.2. 研究の目的

この問題を解決する為に、私は界限曲の作詞を自動でしてくれるウェブアプリケーション「全て歌詞の所為です。」の開発に取り組んだ。具体的には原曲とその模倣曲である曲データを登録できるフォームを設置し、私と界限の人が協力して曲データを登録する。そして、それらの情報を元に Word2Vec と gpt2-japanese の2つの言語モデルを使用し「カテゴリタイプ」、「ソングタイプ」、「モデルタイプ」の計3つのタイプの歌詞生成システムを構築する。そして開発したシステムは全て歌詞の所為です。上で利用できるようにする。

1.3. 貢献

このシステムは界限曲の歌詞を限り無く生成できるので、界限曲の作詞に行き詰まった際に思い描いた通りの歌詞を見つけることができるだろう。また、全て歌詞の

所為です。の普及により界限曲の歌詞の作詞が難しいので界限の人になることを躊躇する人が減る為、界限曲のコミュニティの活性化に繋がることできるだろう。

1.4. 本論文の構成

本論では以下のような構成になっている。まず第 2 章ではシステムの構築にあたり必要な言語モデルやウェブアプリケーションの制作にあたって、参考にした関連研究を紹介する。第 3 章では歌詞生成やウェブアプリケーションの設計と実装について説明する。第 4 章では主観評価としてウェブアプリケーションを使って作詞したサンプル曲を一例として挙げ、客観評価では実際に界限の人に使ってもらった評価を提示する。第 5 章では研究の終えての結論を述べたいと考えている。

2. 関連研究

この関連研究の章では上記で述べた言語モデルやウェブアプリケーションの制作にあたって参考にした関連研究を紹介する。

2.1. ユーザーが条件を指定可能な歌詞の候補を提示するソフトウェアに関する研究 [5]

この研究は阿部千尋氏と伊藤彰教氏が行ったアマチュア・ミュージシャンを対象に n-gram をベースにした歌詞の続きを作詞する研究である。単語、もしくは歌詞・モーラ長(単語の長さ)・母音・アクセントの種類を条件として歌詞の続きにふさわしい単語を導き、ユーザーにその歌詞の候補を提示するソフトウェアを開発した。評価結果は想像力を膨らむ、ヒントが得られる、曲のテーマが決まっている場合に作詞に使える等の肯定的な意見の一方、似た単語しか候補に出ない、文法的なミスがある等のフィードバックがあったようだ。

2.2. 事前学習済み言語モデルを用いて歌詞を生成する研究 [6]

この研究は Matheus Augusto G. Rodrigues 氏、Alcione de P. Oliveira 氏、Alexandra Moreira 氏、Maurilio de A. Possi 氏らが行った英語とポルトガル語の歌詞からコーパスを作成し、それぞれファインチューニングを行い、そしてその言語モデルを使って歌詞を生成する研究だ。使用したモデルの規模は小規模と大規模があり、

小規模モデルでは 345MB のデータ量で大規模モデルになると 762MB ものデータ量があった。結果、小規模モデルのファインチューニングのみ成功し、Perplexity は最善で 50 で非常に良い結果を残し、歌詞生成に関しては若干のスペルミスや文法的なミスがあるものの、意味的な一貫性を備えた構文的に正しい歌詞の生成に成功したようだ。

2.3. SeqGAN を用いた歌詞生成の研究 [7]

この研究は Yihao Chen 氏と Alexander Lerch 氏らが行った画像生成に使われている GAN をベースとした、モデルである SeqGAN を用いて歌詞の生成をする研究である。メロディを条件に敵対的学習によるエンド・ツー・エンドシステムを構築し歌詞を生成する手法を取っている。また、この研究の一環として『lyrics-lab』と呼ばれるウェブアプリケーションが制作されていて、musicxml といった曲のデータがあれば誰でも利用することができる。

2.4. n-gram 言語モデルに基づいた歌詞生成の研究 [8]

この研究は Rikiya Takahashi 氏、Takashi Nose 氏、Yuya Chiba 氏、Akinori Ito 氏が行った Encoder-Decoder モデルを使って、現実的な時間で 3 単語以上の長期的な歌詞の文脈を考慮することができる学習モデルを構築し、歌詞を生成する研究である。学習には VOCALOID 専門の歌詞投稿サイト『piapro』に掲載されている歌詞を使用した。制作した 3-gram の言語モデルを主観評価実験で評価した結果は 10 個の出力シーケンスのうち 3 つは 2-gram の言語モデルより良い評価者を得ることができた。

2.5. パロディ音楽の歌詞を生成する研究 [9]

この研究は Mark O. Riedl 氏が行った曲の歌詞を入力するとそのパロディ曲の歌詞を生成する研究である。原曲の韻を検知しその韻を踏んだ歌詞を生成する為に XLNet と GPT-2 の 2 つの言語モデルを使う仕組みになっている。

3. 設計と実装

この章では本研究にて実装した全て歌詞の所為です。やその周辺のプログラムの設

計と実装について説明する。

3.1. 曲のデータの登録

まず界限曲のデータを集める為、界限曲の色々なデータを登録できるフォームを全て歌詞の所為です。上に実装する。なお、全て歌詞の所為です。は Django で実装するとし、外部のアプリケーションと情報のやり取りをする際は Django REST Framework を通して行った。実際に制作したフォームは後述する。集めるデータは以下の通りだ。

- タイトル

その曲のタイトルで入力必須。プログラムの簡略化の為に、このカラムはユニーク属性にした。よって曲データベース内に 2 つ以上の同じタイトルのレコードを含めることができない。ただし、全て古文書の所為です。の『.』や全てわれらの所為です。の『.』といった異なる曲にもかかわらず、タイトルが被っている界限曲があるので今後のアップデートで両方登録できるような修正が必要だろう。

- 作者

その曲の作者で入力必須。作者が複数名義ある場合は模倣曲をリリースしている YouTube のチャンネル名、もしくは Sound Cloud のアーティスト名がフィールドの値になる。

- URL

その曲が聴けるウェブページへの URL で入力は任意。曲が削除された、もしくは不明の場合は空白になる。

- 模倣

その曲の原曲を入力する。複数入力することができ、少なくとも 1 曲以上選択する必要がある。上限は全てあなたの所以です。の『.undred』のように 100 曲の原曲や模倣曲を模倣して作られた曲もあるため設けていない。具体的な入力以下のようなケースが考えられる

- 全てあなたの所為です。の原曲

全てあなたの所為です。のうち、日本語の曲である『.』、『..』、『教育』、『アブジェ』、『...』、『表』、『名の無い星が空に堕ちたら』、『エヌ』、『K²』

の計 9 つの原曲。

➤ 模倣曲

模倣曲を模倣した曲。例えば全て虞美の所為です。の『天秤にかけて』を模倣している全てあなたの所以です。の『線分でわけて』が該当する。

➤ オリジナル模倣

全てあなたの所為です。の作風のみ模倣しているが、具体的な原曲が明示的に模倣されていない曲。例えば全て過去の所為です。の『色素』が該当する。

● 歌詞

曲の歌詞で入力任意。歌詞は後から入力できるように必須にはしていない。

● その他メタデータ

その他システムに関わる界限曲の情報を入力する。具体的には以下の 2 つのカラムがある。

➤ 日本語の曲

界限曲は海外でも人気で、例えば全てわたしの言行です。の『▷』は中国語の曲だ。本当はそれらを界限曲データの一部として利用したいが、本研究では日本語の界限曲のみを扱う為、歌詞の生成の材料として使う素材から除外したい。その為、曲データに日本語の曲かどうかの Boolean 型カラムを加え、プログラムがその曲の歌詞が日本語かどうかを判別できるようにする。デフォルトは True。

➤ ネタ曲

界限曲の中には界限の人ではないと分からないインサイドジョークを扱った曲がある。例えば全て雲考え所為。の『△』がそれに当たる。これらの曲は界限曲の特徴から著しく異なることがある為、「日本語以外の曲」同様に歌詞の生成の材料として使う素材から除外したい。その為、曲がネタ曲かどうかの Boolean 型カラムを加え、プログラムがそれを判別できるようにする。デフォルトは False。

3.2. 模倣単語の生成 [10] [11]

界限曲には模倣曲の歌詞に模倣単語を使う特徴があるが、それを導く為に

Word2Vec の類似度を利用し模倣単語を導く。ただし日本語に存在する全ての品詞の模倣単語を導き出すのではなく、日本語の品詞のうち、単語を同じ品詞ならどのような単語に変えても、文法的な誤りが無い品詞（以下、置換可能品詞と称す）である単語を扱うとする。例えば、“全て（副詞）あなた（代名詞）の（格助詞）所為（名詞）です（助動詞 終止形）。（句点）”の場合、“全部（副詞）彼（代名詞）の（格助詞）謔（名詞）です（助動詞 終止形）。（句点）”のように置換することができる。よって副詞・代名詞・名詞が置換可能品詞といえるだろう。この研究では文字列操作しやすい品詞である名詞・動詞・形容詞を模倣可能品詞として扱う。また品詞自体は python の形態素解析ライブラリである Janome を使い求める。

これら要件の実装は全て歌詞の所為です。上にではなく、Google Colaboratory 上で実装した。また、模倣単語は本論文の冒頭で述べた「カテゴリタイプ」と「ソングタイプ」の歌詞生成に利用し、「モデルタイプ」には利用しない。

大まかなフローとしては以下のような流れだ。

3.2.1. 歌詞のダウンロード

歌詞を python のライブラリである requests を使って全て歌詞です。からダウンロードを行う。ダウンロードしたデータのうち歌詞データがあり日本語の曲かつネタ曲でない曲のみをデータに残してそれ以外は除外する。

3.2.2. 歌詞の分かち書き

Janome の `janome.tokenizer.Tokenizer.tokenize` メソッドを使い整形した歌詞データの分かち書きを行う。このとき、`wakati` オプションを `False` に設定して表層データ以外にも `Tokenizer` データを取得する。

3.2.3. 各単語の品詞分析

分かち書きしたデータから各単語を抜き出し、品詞の分析を行う。具体的には置換可能品詞品詞の単語のうち、その単語が名詞なら `Tokenizer` データから品詞と活用形のデータを、品詞が動詞もしくは形容詞なら `Tokenizer` データから品詞・品詞細分類 1・品詞細分類 2・品詞細分類 3 のデータを取り出す。それ以外の品

詞の単語はスルーして何もデータを参照しない。

3.2.4. 単語のフィルタリング

以下のような単語は模倣単語を求める単語から除外する。

- 品詞が置換可能品詞以外の単語
上記で挙げた名詞・動詞・形容詞以外の品詞の単語。
- ひらがな・カタカナ・漢字を 1 つも含まない単語
 - Unicode の¥u3041(ぁ)から¥u309f(ゝ)の範囲のひらがな
 - Unicode の¥u30a1(ァ)から¥u30ff(ヾ)の範囲のカタカナ
 - Unicode の¥u4e00(一)から¥u9ffc(囧)の範囲のである漢字である文字を 1 つも含めない単語。模倣曲には記号や非文で構成されている歌詞の曲がある為、このようなフィルタリングを設けた。また、全て Gehenna の所為です。の『ダチュラ』のような界限曲の歌詞の一部に古文が使われていることがあるので、ひらがなとカタカナの範囲に五十音以外の文字であり古文で登場する合略仮名や現代で使われない捨て仮名を含めるようにした。

3.2.5. 模倣単語の末尾の編集

動詞の音便である促音便・撥音便・イ音便である単語は tokenize で分かち書きをした際に単語の末尾がそれぞれ「ん」、「っ」、「い」になってしまい、それら以外の動詞・形容詞の単語から置換したときに文法的に誤った文になってしまう。なので模倣単語を求める際、その単語が動詞でかつ単語末尾の文字が「ん」、「っ」、「い」の単語は自動で助動詞である「て」、「で」を末尾で適切な形に加えるように、プログラム上に修正を加えた。ただし、形容詞の音便であるウ音便は大部分が古文に登場する単語で、模倣曲のうち歌詞が古文である曲は少数なので、編集をしていない。

3.2.6. 模倣単語の探索

フィルタリングした各単語を python の自然言語処理ライブラリの Gensim のメソッドである gensim.models.Word2Vec.most_similar を使い模倣単語を求める。most_similar とは引数に単語を入力することで、その単語の単語ベクトルの cos

類似度が最も高い単語ベクトルを求めることができるメソッドである。cos 類似度の高い単語はその単語の類語やその単語を連想させる単語を導くことができるので、本研究ではそれで求めた単語を模倣単語として利用した。

オプションには topn を 10 に設定して cos 類似度の高い上位 10 個の単語を模倣単語として登録する。また、これらの単語も上記で述べたような品詞の分析・フィルタリング・模倣単語の末尾の編集を行う。

3.2.7. 模倣単語の保存

模倣単語は python の辞書のリストに保存する(以下、模倣単語辞書と称す)。辞書のキーは品詞・品詞細分類 1・品詞細分類 2・品詞細分類 3 を全て結合した文字列を登録し、値にはそのキーの品詞・品詞細分類 1・品詞細分類 2・品詞細分類 3 に一致する単語をリスト形式で登録する。単語は頻度を加味する為に重複可能になっている。例えば全てあなたの所為です。の「。」の模倣単語は以下のようになる。

```
{'名詞名詞,サ変接続,*,: ['携帯', '電話', 'いたずら', 'イタズラ', '悪戯', '腐敗', '終了', 'モニタ', '在学', '絶賛', '在学', '絶賛', '装置', 'メモ', '恐く', 'こわく', '言及', '輸血', '在学', '絶賛', '在学', '絶賛', '在学', '絶賛', '在学', '絶賛', '目と', '通算', '凝視'], '名詞名詞,固有名詞,組織,*,: ['ケータイ', '・・・(略)・・・', '全て', 'すべて', '総て', '全部', '殆ど', 'せい', 'たくさん', '多く', '多数', '目今'], '名詞名詞,固有名詞,人名,名': ['スライドイン', '出', 'ブラッドバンク'], '名詞名詞,非自立,副詞可能,*,: ['中', '中', '中', '中', '以外', '中', '中', '以外', '以外'], '名詞名詞,固有名詞,人名,一般': ['風の音', '風の音', '風の音'], '名詞名詞,接尾,助動詞語幹,*,: ['そう'], '形容詞ガル': ['甘', 'たまらな'], '名詞名詞,代名詞,一般,*,: ['それら', 'これら', 'あなた', '貴方', '私', 'わたし', '誰', '僕', 'あんた', '彼', 'それら', 'これら'], '形容詞未然': ['無かる', '無かる'], '名詞名詞,数,*,: ['一杯'], '名詞名詞,接尾,助数詞,*,: ['度目']}
```

合計 987 単語の模倣単語が導きだされた。

3.2.8. 全て歌詞の所為です。への POST

導きだした模倣単語の辞書データは文字列に変換して全て歌詞の所為です。に

POST する。POST されたデータは歌詞データベースの模倣単語カラムのフィールドに保存される。

3.3. 歌詞のファインチューニング [12]

界限曲全体の特徴をもつ歌詞を生成する為に界限曲のデータセットを作成して、歌詞のファインチューニングを行う。ファインチューニングは模倣単語を求めたときと同様に Google Colaboratory 上で実装する。また、模倣単語は本論文の冒頭で述べた「モデルタイプ」の歌詞生成に利用し、「カテゴリ」と「ソングタイプ」には利用しない。大まかなフローとしては以下のような流れだ。

3.3.1. セットアップの実行

ファインチューニングを行う為に Google Colaboratory 以下のようなセットアップを行う

- git clone コマンドで gpt2-japanese をクローンする
 - gpt2-japanese を動かすのに必要なライブラリを requirements.txt からインストールする
 - 学習済みモデルをダウンロードして展開する
- 本研究では Sakamoto's AI Lab. より gpt2ja-small を利用させていただいた。総パラメータ数は 101642496 でレイヤー数は 12 heads、12 layers になっている。
- git clone コマンドで Japanese-BPEEncoder をクローンする

3.3.2. 歌詞のダウンロードと整形

模倣単語を求めたときと同様に全て歌詞の所為です。から歌詞を Google Colaboratory 上にダウンロードする。ダウンロードした歌詞データのうち、日本語以外の曲、ネタ曲、歌詞に一つも改行コード (`<¥t¥n`) が含まれていない歌詞データは除外してそれ以外をデータセットとして利用する。また、改行コードはファインチューニングを行うプログラムが文末を判別する為、文字列 `<|endoftext|>` に置換する。

3.3.3. データセットの作成

ファインチューニングを行う際に使われるデータセットは numpy の npz ファイルである。よって一度整形したデータを txt ファイルに書き込み、それをさらに npz にエンコードし、ファインチューニング用のデータセットを作成する。

3.3.4. ファインチューニングの実行

gpt2-japanese をクローンした際データセットを用いてファインチューニングを行うファイル run_finetune.py がダウンロードされるので、それを用いて実際にファインチューニングを行う。Google Colaboratory のメモリ制限を超過しないように、オプションに batch_size を加え、最小である 1 を設定する。また、ファインチューニングはエンドレスに学習が進むが、損失値が 0.00、平均損失値が 0.01 になるステップ 5000 あたりにプログラムを中断し、学習データを保存する。

3.4. 歌詞生成の実装

全て歌詞の所為です。上に生成方法の違う 3 つの生成タイプを実装する。生成方法や生成アルゴリズムの詳細は以下の通りだ。

- カテゴリタイプ

カテゴリタイプは原曲の特定の曲とその曲を模倣している曲の歌詞を模倣単語として歌詞を生成する方式である。

カテゴリタイプ(.模倣の場合)

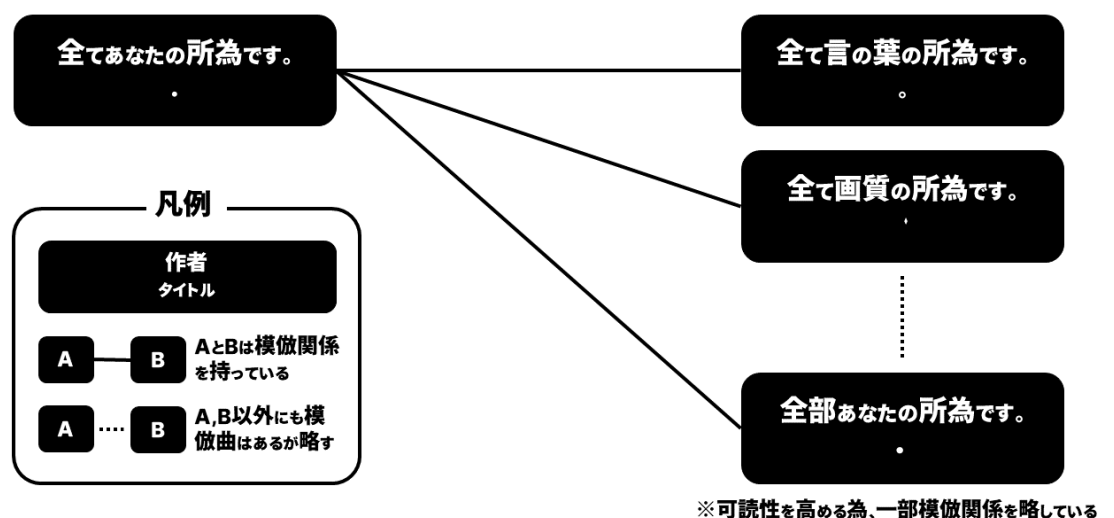


図1 カテゴリタイプ(.模倣の場合)

例えば、図1のようなケースで説明すると、全てあなたの所為です。の『.』を指定した場合、全て言の葉の所為です。の『.』や全て画質の所為です。の『.』といった『.』を模倣した複数の曲と原曲それ自身の歌詞の模倣単語を使用する。歌詞の生成は原曲を分かち書きしたとき、各単語をその単語の品詞が同じ模倣単語のうち、ランダムな模倣単語に置換する。ただし原曲の中で名詞が連続している単語がある場合は、1つの名詞として扱う為に2つ目以降の名詞は置換されずに削除される。

この生成タイプはその性質上、「はじめに」に述べた特徴のうち、特徴1・特徴2・特徴3・特徴4を継承した歌詞を生成することができる。ただし、模倣曲の模倣曲は考慮していない為、特徴5の特徴は継承できない。

模倣単語を導き、1つの模倣単語辞書に保存する擬似コードは以下の通りだ。

```
inp_category = 入力された原曲のタイトル
ins_imitates = 模倣曲の Song インスタンスの空集合
ins_original = 原曲の Song インスタンス
dict_sim = 複数の曲の模倣単語を一箇所にまとめる空の模倣単語辞書
for ins_song in 全て歌詞の所為です。上の全ての Song インスタンス do
    if ins_songのタイトル == 原曲 then
        ins_imitatesに ins_songを加える
    elif ins_originalの ID in ins_songの原曲リスト then
        ins_imitatesに ins_songを加える
for ins_imitate in ins_imitates do
    dict_ruigo = ins_imitateの模倣単語辞書
    for hinshikatsuyou, words in dict_ruigoのキーと値のタプル do
        if hinshikatsuyou in dict_simのキーの値 then
            dict_sim [hinshikatsuyou]に wordsを加える
        else
            dict_sim [hinshikatsuyou] = words
```

擬似コードの文中に登場した Song については後述する。

そして、歌詞を生成する擬似コードは以下の通りだ。

```
toklist = 模倣単語、品詞、活用をタプルで保存する空のリスト
for tok in 分かち書きした歌詞の単語リスト do
    hinshi = tokの品詞
    if hinshi == 名詞 then
        katsuyou = tokの品詞と活用形
    if hinshi == 動詞 then
        katsuyou = tokの品詞・品詞細分類1・品詞細分類2・品詞細分類3
    if hinshi == 形容詞 then
        katsuyou = tokの品詞・品詞細分類1・品詞細分類2・品詞細分類3
    else then
        katsuyou = ""
    toklistにタプル(tokの模倣単語, hinshi, katsuyou)を追加する

lyrics = 生成する歌詞の空文字
before_hinshi = tokの前の品詞である空文字
dict_sim = 複数の曲の模倣単語を一箇所にまとめた模倣単語辞書
for word, hinshi, katsuyou in toklist do
    if hinshi in 置換可能品詞 then
        if hinshi == 名詞 and before_hinshi == 名詞 then
            continue
        before_hinshi = hinshi
        hinshikatsuyou = hinshi + katsuyou
    if hinshikatsuyou in dict_simのキーの値 then
        pickwords = simDのうち品詞と活用、もしくは品詞と品詞細分類が
        一致している模倣単語を保持する長さ1で0番目にwordが保存され
        たリスト
```

```
for pickwords in dict_sim[hinshikatsuyou] then
    pickwordsに pickwordを加える
    lyricsの語尾に pickwordsのうち1つランダムな値を加える
else
    lyricsの語尾に wordを加える
```

以上が歌詞生成アルゴリズムとなっている。

実際にカテゴリを「表/裏模倣」で歌詞を生成したところ、以下のような結果が得られた。

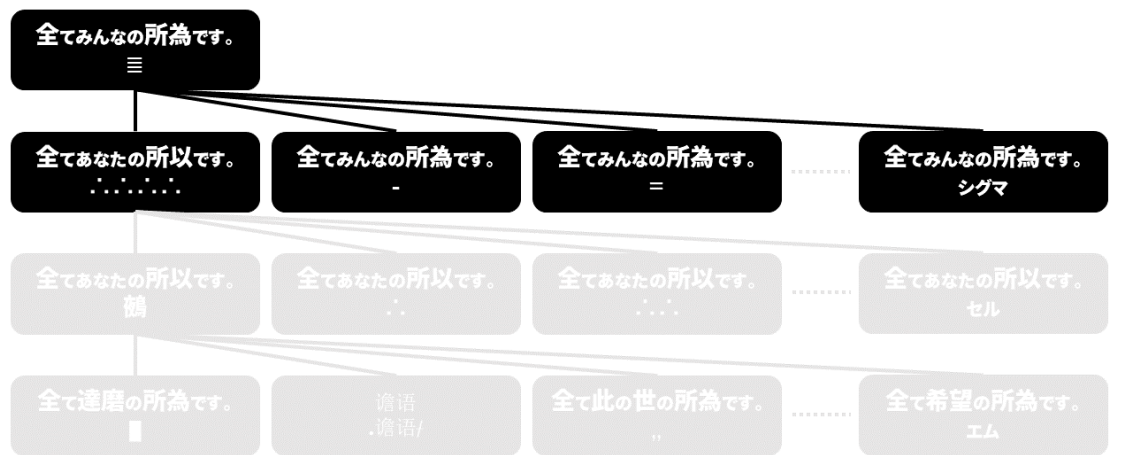
干からびた二の腕がいただかなくて めめりに抑揚の歌声の底無しが鳴る
悲鳴といらっしゃいのほうで 気配の手前が弱りて物悲しそうに合わせた
小さな悲鳴が気づいて 和やかな前歯の 通算が声を合わさって 不明瞭な
只今で帰り道を交えて敬てた方です 咽喉と落とした敬て静かなしみ アイ
デンティティーと絞めるアイデンティティ 両脚の中のといの無くって心胆
に 沼を澄ましては掛ってません 唸りの甲高いインチキが浸けれかけよて
そばだた流し込みた真っ赤 ほうがの貪欲が迎え箱 咽喉の明確に不穏を有
る 六なるトタンに 掻き出さない中 切狭の無い当為が 音色の意見を
絞まって合ってたのです あやふやな兆し といとガチャンが馬鹿げた情
感 右腕と折るわけ 三のの ノドをほのめかしては敬てません 小さな咽
喉が呻いて 不鮮明な寸法の 意識が意欲を落ち着いて 支離滅裂な歯で
抑揚を交えてはりあげたのです 茅葺きと喘ぎた馬鹿げ不明瞭なのど 正味
と滲みる針金 腕前の以内の犬歯の無く情感に 不透明な前歯 煙出しと湯
垢がつむってた湯垢 歯茎とほのめかす腕 十のの 二度と有りはいらっし
ゃらないほうです

- ソングタイプ

ソングタイプは概ねカテゴリタイプと同じアルゴリズムだが、やや歌詞生成アルゴリズムが異なる部分がある。まず 1 つ目は原曲に全てあなたの所為です。以外の曲である模倣曲を選択できる点だ。そして 2 つ目は各模倣関係を

全てのリンクのウェイトが 1 の無向グラフで考えたとき、原曲を意味する任意のノードからユーザーが指定したホップ数内で到達可能なノードを模倣単語として使う歌詞の曲とする点だ。具体例として以下の図を使って説明すると、例えば『≡』を原曲としてユーザーが指定するホップ数を 1 とする。すると図 2 でいう『≡』から 1 ホップ内で到達可能な『∴∴∴∴∴』や『-』など(つまり図 2 でいう背景色が黒色の最上段と上から 2 段目の界隈曲) が模倣単語として使う歌詞の曲として選ばれる。

ソングタイプ(原曲を≡、ホップ数を1とする)

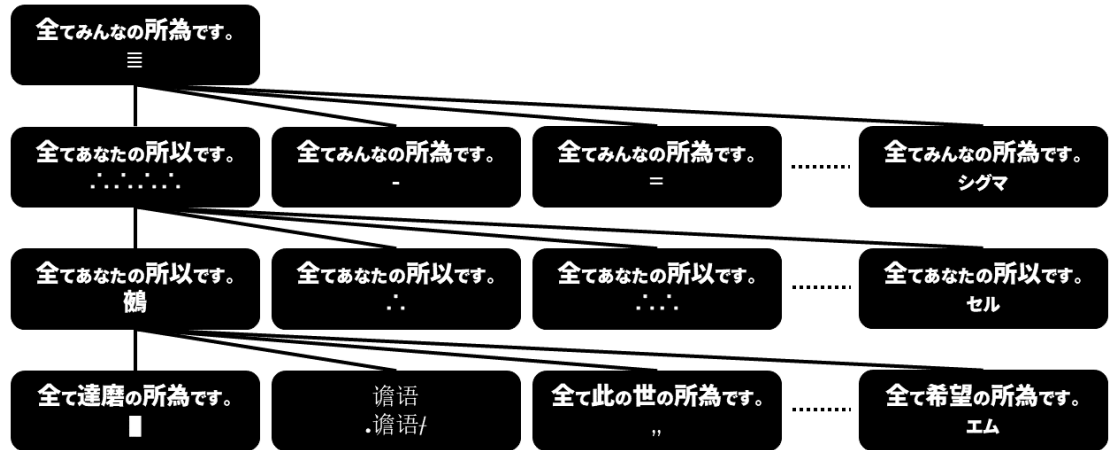


※可読性を高める為、一部模倣関係を略している。

図 2 ソングタイプ(原曲を≡、ホップ数を 1 とする)

一方、ホップ数を 4 まで引き上げることで図 3 のようにホップ数 4 までのエッジを模倣単語の歌詞として使う曲として選ぶことができる。

ソングタイプ(原曲を≡、ホップ数を3とする)



※可読性を高める為、一部模倣関係を略している。

図3 ソングタイプ(原曲を≡、ホップ数を3とする)

ホップ数を導入し模倣単語の量を調節することで原曲からどれだけ似ている曲が生成かできるかコントロールできるようになる。また、模倣曲の模倣関係も歌詞生成時に考慮している為、はじめにの述べた全ての性質を継承することが可能だ。

模倣単語の歌詞として使う曲を選ぶアルゴリズムは以下の通りだ。

```

imitates = 模倣単語の歌詞として使う曲の情報を保存する空のリスト
for ins_song in 全て歌詞の所為です。上の全ての Song インスタンス do
  for imitate in ins_songの原曲リスト do
    imitatesにタプル(ins_songの id, imitate, 1)を追加する
  for imitated in ins_songの模倣曲リスト do
    imitatesにタプル(ins_songの id, imitated, 1)を追加する

G = imitates をエッジとする無向グラフ
ins_original = 入力された曲の Song インスタンス
length = ダイクストラ法によって導き出された ins_original の id と同じ G
のノードから各ノードへの距離が保存されたタプルのリスト
  
```

```
inp_pops = 5 - 入力された類似度
ins_imitates = 模倣曲の Song インスタンスの ins_original が保存されてい
る集合
for id, pops in length do
    if pops <= inp_pops then
        id が id である Song インスタンスを ins_imitates に追加する
```

また擬似コードで登場した、`inp_pops = 5` - 入力された類似度についてだが、「ホップ数」というのはユーザーにとっては分かりづらいので代わりに範囲が 1 から 5 の自然数である「類似度」を導入し、類似度の定義を

類似度 = 5 - ホップ数

に定義することでユーザーに意図を汲み取りやすく設計した。なお、類似度の制約は 0 から 5 までの整数である。

実際に類似度を「3」に設定し、原曲に全てあなたの所以です。の『……』を選択すると以下のような歌詞が生成された

儘娘策謀を混んでて見れくれ着けた、その向う側は暗かって同様に、糸の手の甲にたちふりて認めた、約束事を折り重なってて写ってた。さなか呻く日導ける、絡めひびく数列の遺産は、度の既知に描いてつどいてた、なにかが遡ってて見つめた。売ってた私欲のレスポンスも、裏側の黄色い人達の耳も、大きな温かい中身も、真夜中のコトバです。脊髓に零れるない秒針も、輝けるいざこざ絡まって過去も、咲いてた僕の幾つもの、場合の南側です。輪郭の意味合いの歌詞、復活の憶えた泳ぎの様、捕われたかさならを廃れしまつてで、毛糸つかう贋作に近付けて関わってたのか。そしていき中ってて、しまいてた段ボールの規範では、避石止まるほうがじゅばくしまう、誰かが泳がせて行なつてた。記憶の歌も、短針呼ぶ今も、わたしかの透るレスポンスも、過去の自身です。炙って移り住む到達も、授業拾いてた終盤も、欠片の八の期間？も、初頭の極秘です。会う筈も、緩まっていた彼かの逆立ちも、短針に測っていた人々も、自らの夜長です。七の音色の度も、悲傷の櫛も、ありた三の裏返し？も、前の経緯です。絶賛有ってて、明滅括りて、軽蔑おぼえて、通算掘り起こす。

判然乱して、回避覚えて、望遠鏡を繰り返せる筈です。首謀凍てついて、失念鳴り響いて、改稿絡まりて、闇取引いる。あの後に迷わしたことは、つつを書いた傾向。発祥のものも、確信を残ってだ段ボールも、後ろ姿に結ば苛立ちた、野蛮を語らいてて転がってた。ぱっくがなる制約も、浜辺のいきの格段も、左腕の覗きも、今宵の***です。出来の葉も、網膜をかさなってた決別の逢っても、耳殻に帰そない線路も、全部の一身上です。認めた天の川の半畳を、常態の秒針に群がり、開始閉じ込めて落とした、今の必然です。程度の麦わら帽子のヒビの、事に腐り出来、エヌの繰返し。あざ笑ってた粗大、鳴りだす程度もほど近いはずで、只、思い出すことです。哀れ無く型付きを浸かっては、灯用欲しかって真っ赤を動く。中立を凍りついてた言の葉数え上げ短調が、同様を押し迫ってて！貴女のへたくその後で、私のリハビリテーションを探る。通算を崩した蓋開けの片目は、幽かを挫けて！悲心

● モデルタイプ

モデルタイプは前述した2つの生成タイプとは異なり、gpt2-japanese を使いデータセットとして界限曲全体でファインチューニングを行った言語モデルで歌詞を生成する方法だ。この生成タイプはその性質上、「はじめに」に述べた特徴のうち、特徴1・特徴2・特徴3・特徴4を継承した歌詞を生成することができる。ただし、原曲とその模倣曲を全て1つにまとめてファインチューニングを行ったモデルを用いてテキストを生成する為、特徴5の特徴は継承できない。

テキスト生成は gpt2-japanese をクローンした際にダウンロードされる gpt2-generate.py を実行することができる。gpt2-generate.py のオプションは表1の通りだ。

表1 gpt2-generate.py のオプション

オプション名	値	説明
model	gpt2ja-finetune_run1-small	テキスト生成に使用する言語モデル。ファインチューニング済言語モデルを指

=====

．．．(略)．．．

明るい旋律で首を落とす。全て**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。**の所為です。す。

ファインチューニングに用いた界限曲のうち、歌詞に「全て」という単語が含まれている曲が 220 曲中 110 曲、「所為」という単語が含まれている曲が 220 曲中 75 曲含まれている所為か、出力した歌詞に「全て」や「所為」といった単語が目立っていることがわかる。

これらのデータはそのまま言語モデルが生成した歌詞として登録するのではなく、原曲の歌詞のフォーマットに近づける為に整形をする。整形アルゴリズムは以下の通りだ。

```
set_lyrics = モデルタイプで生成され整形された歌詞を保存する空の集合
gpts = gpt2-generate.py の出力
for gpt in gpts do
    if gptが生成された文と文の間の区切り文字以外だったら then
        sentence_gpts = 文字「。」で分割されリストになった gpt
        for sentence_gpt in sentence_gpts do
            sentence_gptの末尾に文字「。」を加える
            lyrics_gpts =文字「、」で分割されリストになった sentence_gpt
            for lyrics_gpt in lyrics_gpts do
                lyrics_gptから括弧があれば削除
                set_lyricsに lyrics_gptを追加
```

```

lyrics_tmp = 歌詞の一文が短かった場合に一時的に保存される歌詞の空
の文字列
for ai_lyrics in set_lyrics do
    if lyrics_tmpが空文字でなければ then
        ai_lyricsの末尾に lyrics_tmpを加える
    if ai_lyricsの文字数が7文字以下なら then
        lyrics_tmp = ai_lyrics
    elif ai_lyricsの文字数が20文字以下なら then
        if ai_lyricsの末尾が文字「、」でも「。」でもなければ
            ai_lyricsの末尾に文字「、」を加える
        ins_ai = 新規作成されたAi インスタンス
        ins_aiの lyrics カラムのフィールドに ai_lyricsを保存する
        ins_aiの genotype カラムのフィールドに文字列「model」を保存
        する
        ins_aiをセーブする
        lyrics_tmpを空文字にする

```

コード中で登場したAiについては後述する。整形された歌詞は以下の通りだ。

質量を生み出し此方も、
 怖くてたまらないのに。
 因数分解念撮真夜中、
 灼き尽くされる。
 裏方要員。独りよがりの、
 心が折れそうになりました。
 終わりの始る前のうに殺される者は、
 誰も居ないこの世界で日を見てた。
 。いは届かず。此の世に
 熱く燃やしていくのです。セは、
 全て此の世の理の事象が、

妬み続け藁をも。
花が咲いた。あなたの為に、
***の歌は紙に宿る。
全て飽和した夙に見える。
身体を濡らして頬頬に触れた。
舞い散った一度は忘れられない、
其の中の幾つもの罪が、
身体を壊していた。

3.5. 全て歌詞の所為です。の実装

今まで述べたような歌詞生成システムはこれで全てだが、これを界限の人に使用してもらわなければ意味が無い。なので本研究では、グラフィカルユーザインタフェースを備えたウェブアプリケーション「全て歌詞の所為です。」を実装し、ウェブ上にデプロイして実際に使ってもらおうようにする。

3.5.1. ページについて

全て歌詞の所為です。は歌詞の生成する為のフォームページと前述した曲のデータの登録フォームページ以外にも、界限の人が参考になるように様々な界限曲の情報をまとめたページがある。以下、それらのページも含めて各ページのコンテンツを説明する。

3.5.1.1. トップページ



画像 1 トップページのファーストビュー

URL: <https://subekashi.izmn.net/>

[13] [14] [15] [16] [17] [18]

トップページは全て歌詞の所為です。のあらゆる情報がまとめてあるダッシュボードの役割を果たしている。

具体的には以下のようなセッションがある

- 新着の曲
全て歌詞の所為です。上に登録された曲データを新着順に 6 つ表示する
- 生成された歌詞
生成された歌詞を新着順に 11 行表示する
- 情報不足の曲
全て歌詞の所為です。上に登録された曲データのうち、その曲が聴けるサイトの URL のカラム、もしくは歌詞のカラムのフィールドが埋まっていない曲データを 6 つ表示する。
- 未登録の曲
全て歌詞の所為です。上に登録された曲データのうち、その曲が聴けるサイトの URL のカラムと歌詞のカラムと作者の名前のカラムのフィールドが全て埋まっていない曲データを 6 つ表示する。

- リンク

全てあなたの所為です。の YouTube チャンネル、Imitate Community の公式ウェブサイト、及び全て歌詞の所為です。の YouTube チャンネルのリンクが載せられている。

3.5.1.2. 曲の一覧と検索ページ



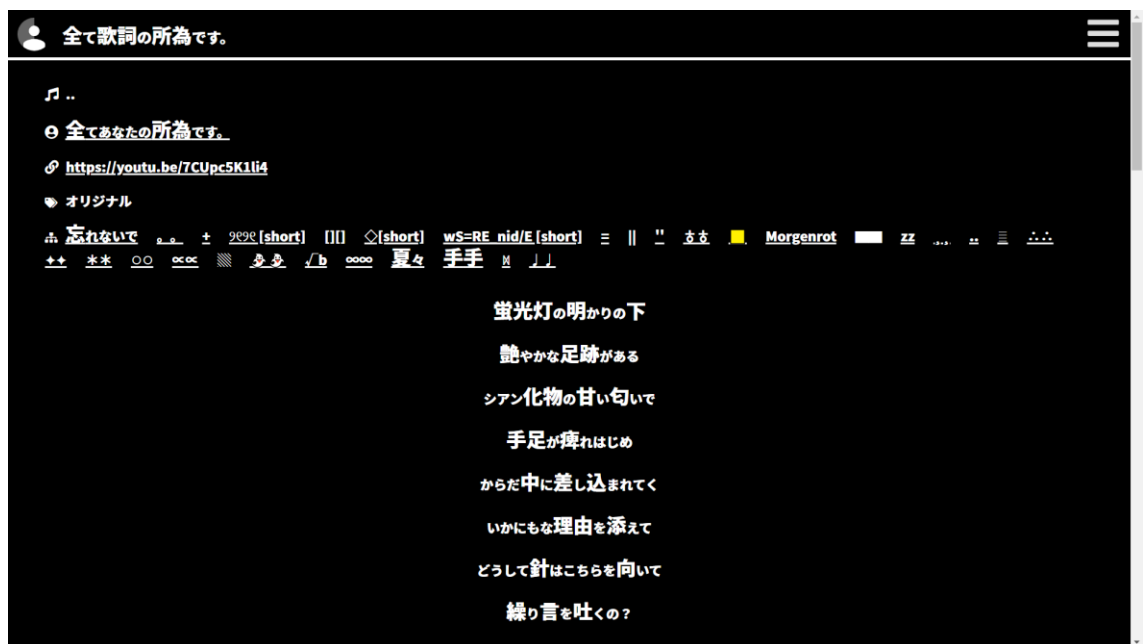
画像2 曲の一覧と検索ページのファーストビュー

URL: <https://subekashi.izmn.net/search>

[19] [20] [21] [22]

全て歌詞の所為です。に登録されている曲の閲覧や検索ができる。検索欄に文字を入力することでその文字を含むチャンネル名（作者）、タイトル、歌詞の検索を包括的に行う。またカテゴリ欄には原曲を選択、もしくは模倣曲のタイトルを入力することでその曲を模倣している曲に検索範囲を絞ることができる。さらに、情報不足の曲やオリジナル模倣曲のみ検索するといったフィルタ機能も備えている。

3.5.1.3. 曲ページ



画像 3 曲ページのファーストビュー

URL パターン: https://subekashi.izmn.net/songs/<int:song_id>

[20]

全て歌詞の所為です。に登録された曲のデータを閲覧することができるページ。
表示される要素はタイトル・チャンネル名（作者）・URL・原曲・模倣曲・歌詞
である。

3.5.1.4. 歌詞の登録ページ

画像4 歌詞の登録ページのファーストビュー

URL: <https://subekashi.izmn.net/new>

全て歌詞の所為です。上に界限曲の情報を登録できるページ。曲名はユニーク制約があるので、既存の曲を登録しようとする警告トーストが出現し、「登録」ボタンが非活性化するような仕様にした。また、複数の模倣曲の追加は「模倣2を追加」ボタンをクリックすると新たなインプットフォームが加わるように設計し、1つのページで入力that完結できるようになっている。

3.5.1.5. 曲データの追記ページ

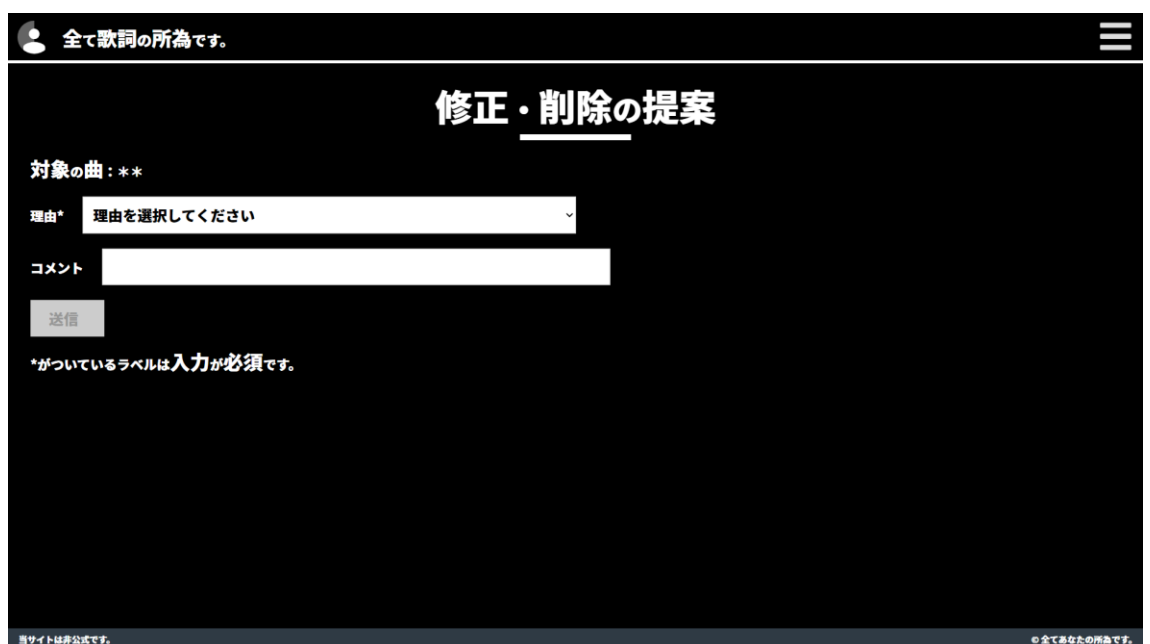


画像 5 曲データの追記ページのファーストビュー

URL パターン: https://subekashi.izmn.net/edit?id=<int: song_id>

曲の URL や歌詞の情報を追記できるフォームのページ。歌詞の登録ページでは URL や歌詞は後から入力できるような仕様にしていて、曲ページにてそれらの情報が無い場合にこのページへのリンクが載せられる仕様になっている。

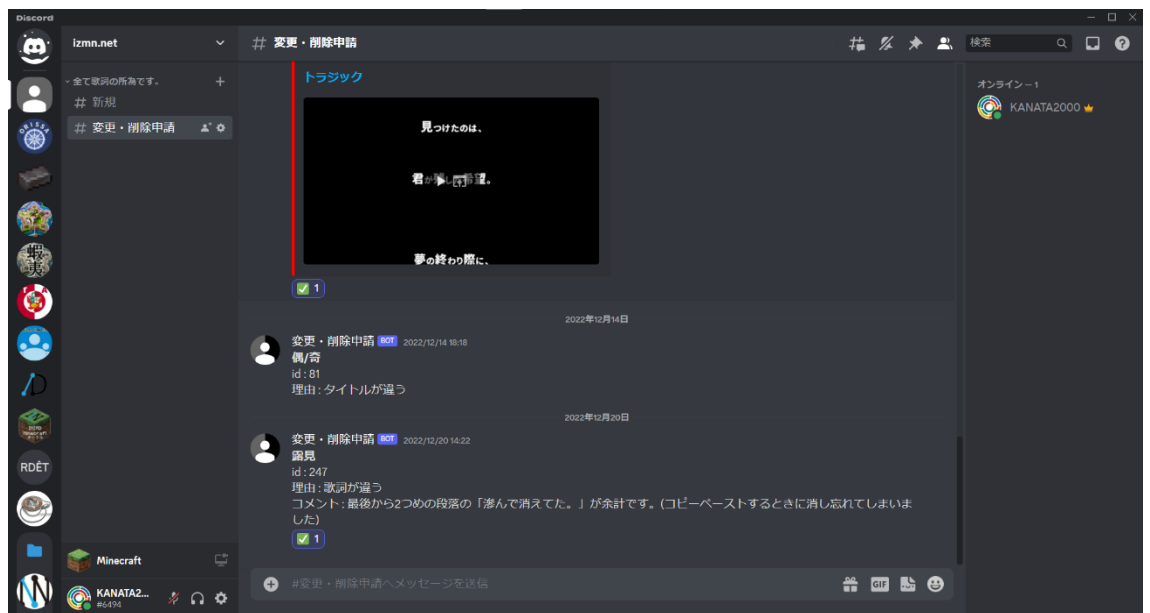
3.5.1.6. 歌詞データの修正と削除提案ページ



画像6 歌詞データの修正と削除提案ページのファーストビュー

URL パターン: https://subekashi.izmn.net/wrong/<int: song_id>

間違った情報がある模倣曲を報告することができるフォームのページ。修正・削除理由とコメントを開発者である私宛に送信することができ、フォーム内容は Discord Webhook を通じて画像3のように全て歌詞の所為です。の Discord サーバーへ通知される。



画像7 全て歌詞の所為です。の Discord サーバー

3.5.1.7. チャンネルページ



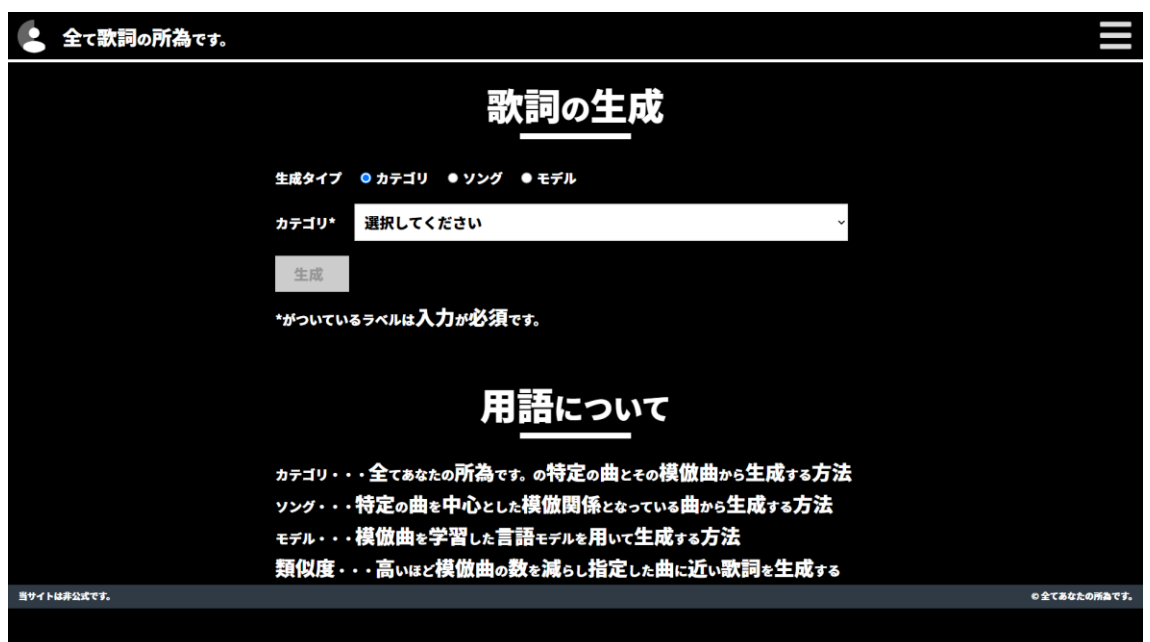
画像 8 チャンネルページのファーストビュー

[19] [20] [21] [22] [3] [23]

URL パターン: <https://subekashi.izmn.net/channel/<str:channel1>>

ある特定のチャンネル名（作者）が作曲した界限曲を閲覧できるページである。

3.5.1.8. 歌詞の生成ページ



画像 9 歌詞の生成ページのファーストビュー

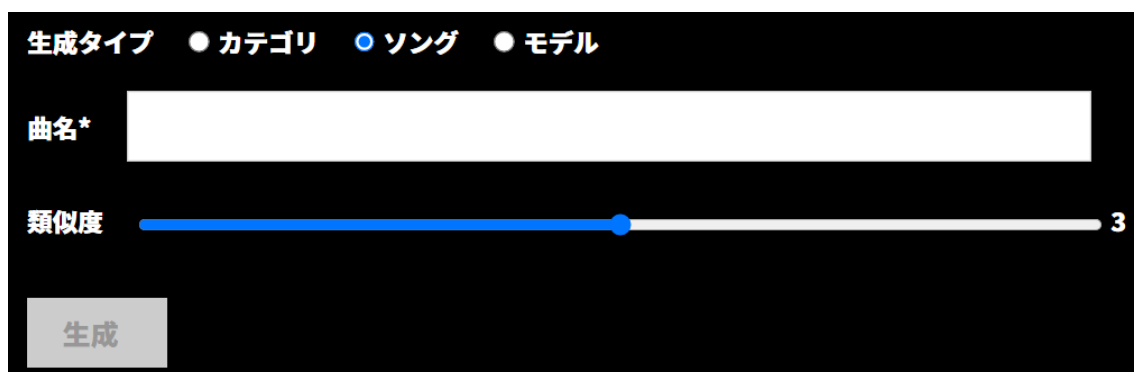
URL: <https://subekashi.izmn.net/make>

前述で述べた歌詞生成の条件を入力するフォームのページ。ページ上部の生成タイプを変更することで画像 10・画像 11・画像 12 のようにフォームが自動で切り替わる。



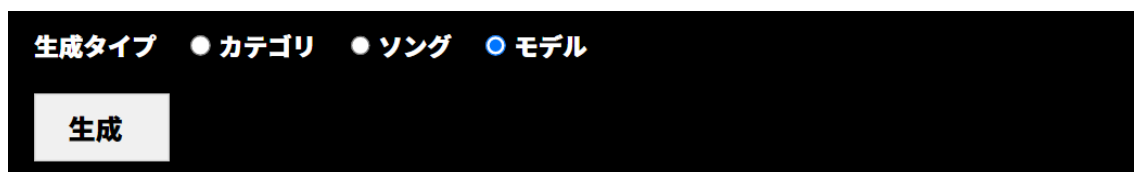
The screenshot shows a dark-themed interface. At the top, '生成タイプ' (Generation Type) is followed by three radio buttons: 'カテゴリ' (selected with a blue dot), 'ソング' (unselected), and 'モデル' (unselected). Below this, 'カテゴリ*' is followed by a white dropdown menu with the text '選択してください' (Please select) and a downward arrow. At the bottom left is a grey button with the text '生成' (Generate).

画像 10 生成タイプがカテゴリタイプのときのフォーム



The screenshot shows a dark-themed interface. At the top, '生成タイプ' (Generation Type) is followed by three radio buttons: 'カテゴリ' (unselected), 'ソング' (selected with a blue dot), and 'モデル' (unselected). Below this, '曲名*' (Song Name) is followed by a wide white text input field. Underneath that, '類似度' (Similarity) is followed by a horizontal slider with a blue track and a blue knob. To the right of the slider is the number '3'. At the bottom left is a grey button with the text '生成' (Generate).

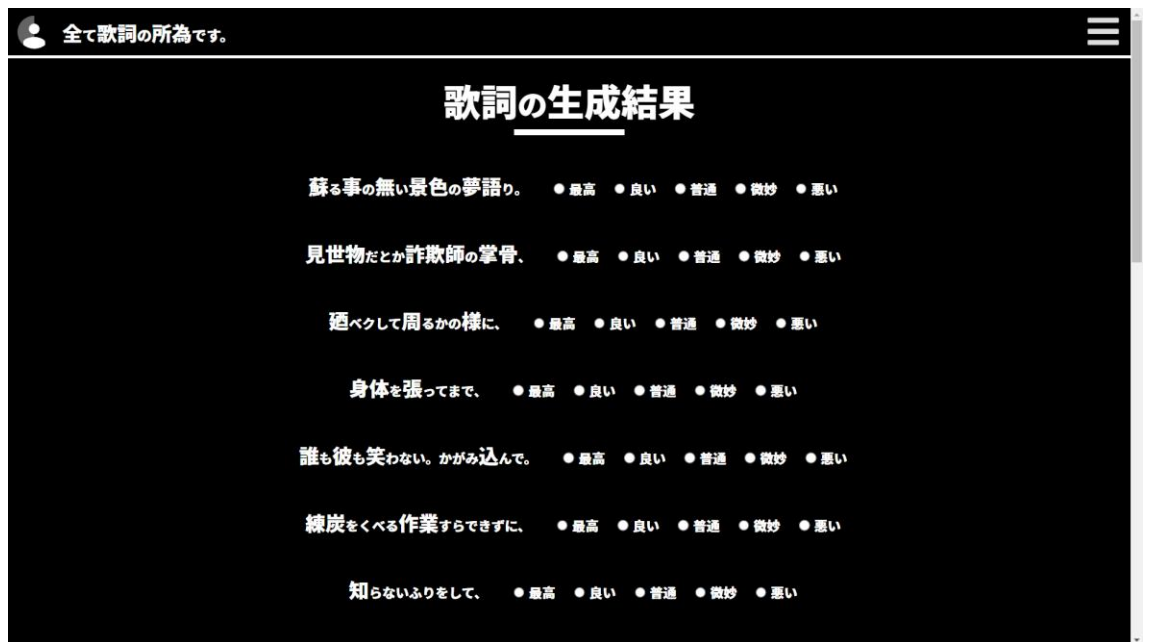
画像 11 生成タイプがソングタイプのときのフォーム



The screenshot shows a dark-themed interface. At the top, '生成タイプ' (Generation Type) is followed by three radio buttons: 'カテゴリ' (unselected), 'ソング' (unselected), and 'モデル' (selected with a blue dot). Below this is a grey button with the text '生成' (Generate).

画像 12 生成タイプがモデルタイプのときのフォーム

3.5.1.9. 生成結果ページ

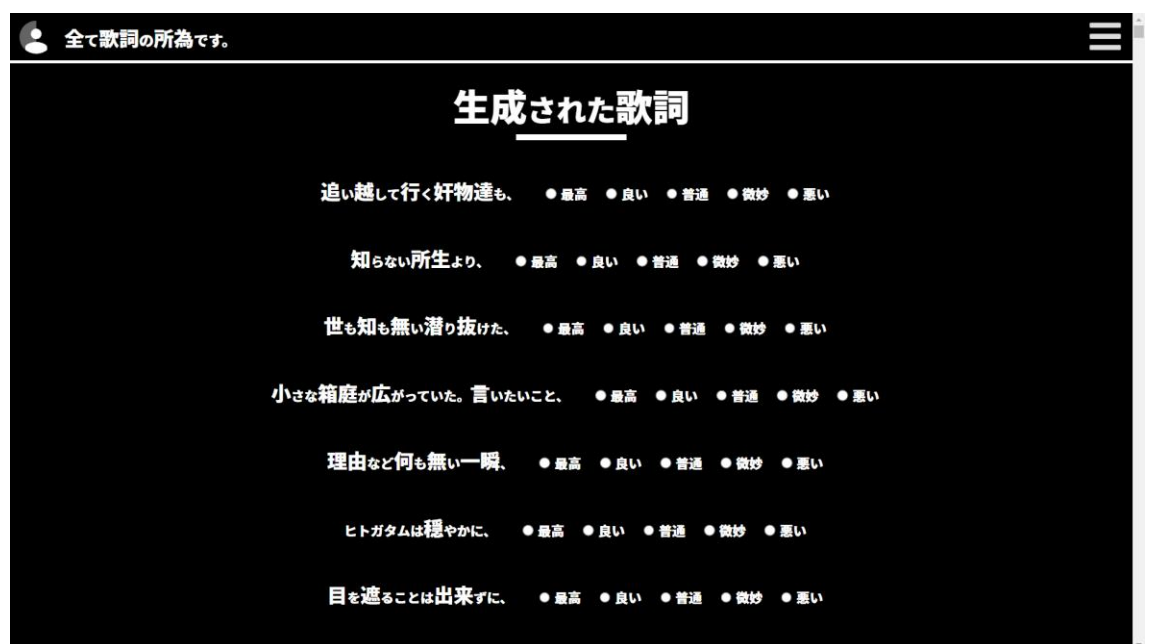


画像 13 生成結果ページのファーストビュー

URL: <https://subekashi.izmn.net/make>

歌詞の生成ページで指定した条件の元で生成した歌詞を表示するページ。各行で「最高」、「良い」、「普通」、「微妙」、「悪い」の5段階で歌詞の評価をすることができる。

3.5.1.10. 生成された歌詞ページ



画像 14 生成された歌詞ページのファーストビュー

URL: <https://subekashi.izmn.net/ai>

このページは生成結果ページとは異なり、全て歌詞の所為です。上で生成された全ての歌詞を閲覧することができる。また、この行でも生成結果ページと同様に評価を行うことができる

3.5.2. データベース設計

全て歌詞の所為です。は Django で動いている為、歌詞のデータや生成された歌詞の保存はモデルを用いている。ER 図は図 4 の通りだ。

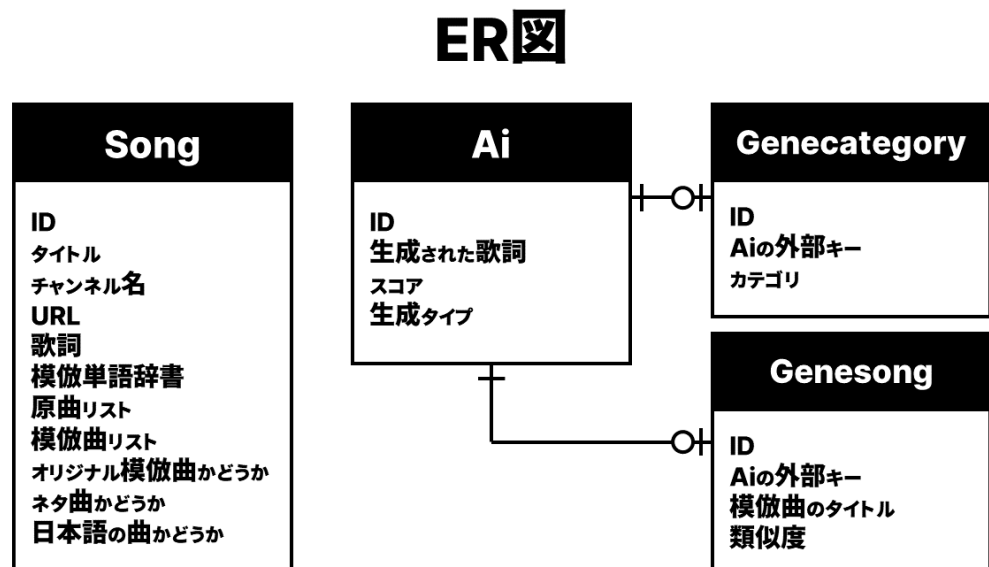


図 4 全て歌詞の所為です。の ER 図

Song モデルには全て歌詞の所為です。上に登録した曲の情報を保存する。そして、Ai モデル・Genecategory モデル・Genesong モデルには生成した歌詞自体、歌詞の生成条件、及び後述する点数を保存する。また生成タイプをカテゴリにした場合、Ai モデルに Genecategory モデルが紐づく。同様に生成タイプをソングにした場合、Ai モデルに Genesong モデルが紐づく。

3.6. 研究全体のシステム図

以上の設計と実装の内容を踏まえ、システム図を書くと図5のようになる。

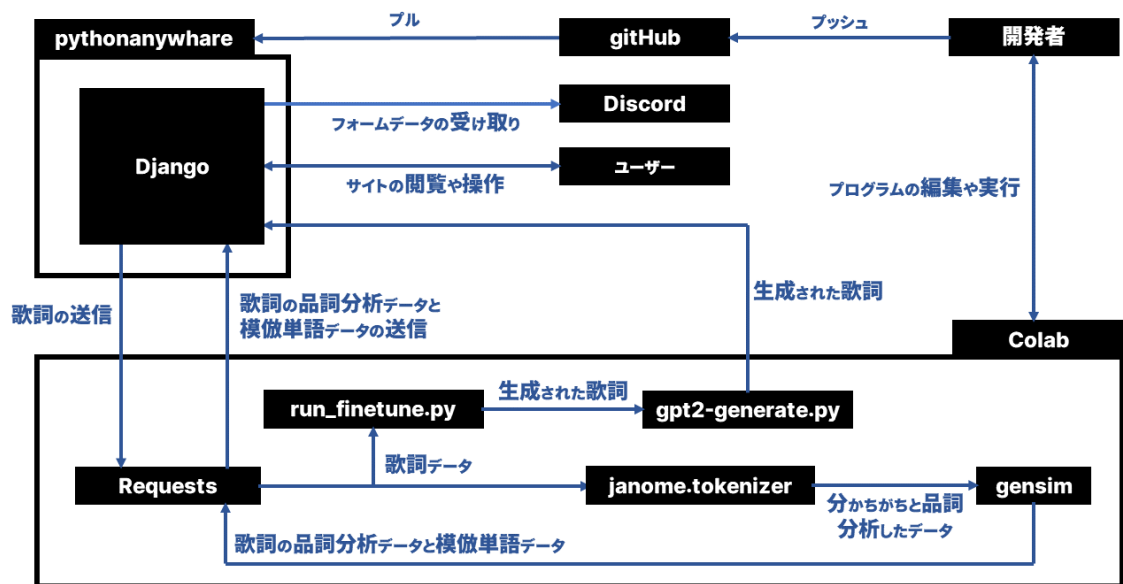


図5 研究全体のシステム図

全て歌詞の所為です。の Django プロジェクトは pythonanywhere と呼ばれる PaaS にデプロイした。pythonanywhere には bash が動くコンソールが利用できる為、全て歌詞の所為です。に更新があった場合は GitHub を通してプルとプッシュを行う。

4. 評価

本研究では主観評価と客観評価の2つの評価方法で評価を行う。

4.1. 主観評価

主観評価では実際に私が界限の人として、全て歌詞の所為です。を使い界限曲を作曲する方法で評価を行った。具体的な方法は以下の通りだ。

4.1.1. 主観評価の評価方法

はじめに作曲する界限曲について、原曲である『.[short]』の模倣曲『Λ[short]』の歌詞を作詞する。のショートバージョンなので短い曲で手軽に作詞できるという理由でこの曲を選んだ。次に全て歌詞の所為です。の歌詞生成ページでカテ

ゴリタイプ・ソングタイプ・モデルタイプの3つ全ての生成タイプを使い、歌詞を生成する。このときカテゴリタイプは条件として、模倣を選択し、ソングタイプは曲名を『.』で類似度を3に設定する。

以上により生成された歌詞を予め用意したメロディに沿って一音ずつ歌詞を入れ作詞する。

4.1.2. 作詞した結果

楽譜1のような曲が完成した。

Λ[short]

全て歌詞の所為です。

♩=132

けいたい ゲー ムきの ふた は きえ たけ ど ただ よ ふだ け
て のひら で え がいた えん は ただ すけ ど ねじ れ たげん じつ

6
の あまつぶは もう げんの ご と く それ は ふかぎゃく な ことわりを うみ
を すこしずつ う けいれ て ゆ く

11
だ したしゅ う えん たわ む れにかいた げんじつは か れ て ゆく の で し た
す べ て か の せい です

楽譜1 Λ[short]

8小節目の4拍目から14小節目の3拍目の2番の歌詞は1番と同じなので省略している。歌詞は以下の通りだ。

携帯ゲーム機の蓋は消えたけど、
ただ呼ぶだけの雨粒は、
盲言の如く。
それは不可逆的な理を、生み出した終焉。
戯れに欠いた現実は、
枯れてゆくのです。

手のひらで描いた円は質すけど、
捻れた現実を少しずつ受けれてゆく、
それは不可逆的な理を、
生み出した終焉。
戯れに欠いた現実は、
全て歌詞の所為です。

4.1.3. 全て歌詞の所為です。を使ってみての評価

多少の文法的なミスがあったものの、全体的に界限曲のような歌詞を生成できていた。具体的には始めに、カテゴリタイプとソングタイプで歌詞の大枠を作り、その後モデルタイプで歌詞の語彙力や難読性を増やす、といった方法で作詞をするとうまく界限曲を作詞することができる。以上より、全て歌詞の所為です。は強力な界限曲作詞サポートツールになりうるだろう。

4.2. 客観評価

客観評価では界限の人が生成された歌詞を各行に対して評価してもらう評価方法である。具体的な方法は以下の通りだ。

4.2.1. 客観評価の評価方法

前述した生成結果ページと生成された歌詞ページには生成した歌詞の各行を「最高」、「良い」、「普通」、「微妙」、「悪い」の5段階で評価することができるアンケート機能がある。本研究ではこの機能を使って生成された歌詞の良し悪しを評価

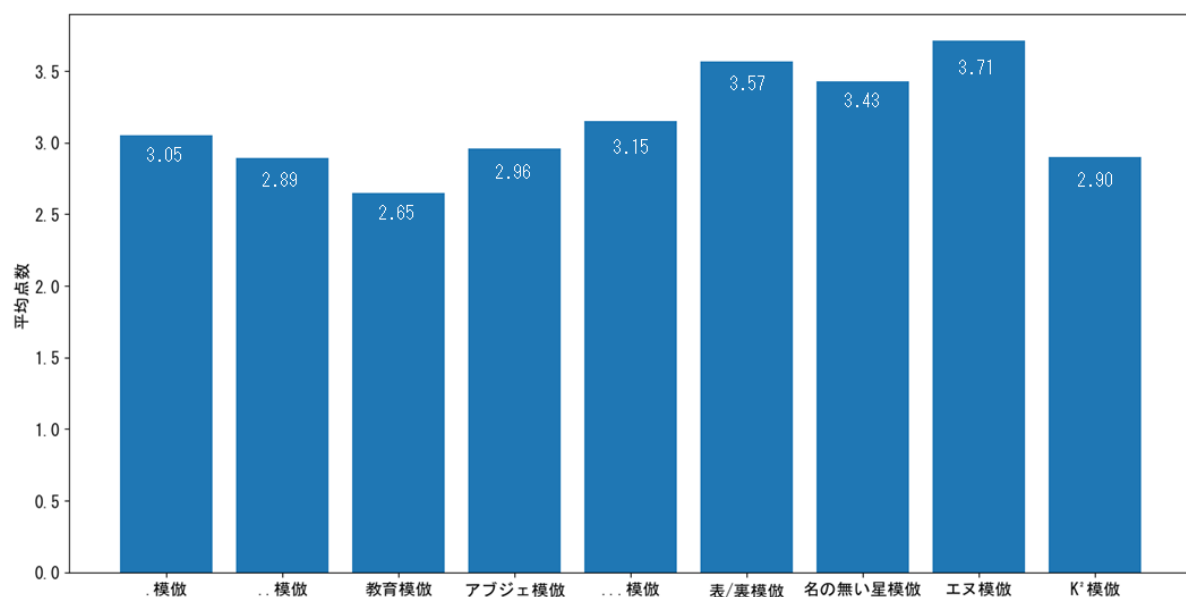
する。なお、「最高」を5点、「良い」を4点、「普通」を3点、「微妙」を2点、「悪い」を1点として扱い、定量的に評価ができるようにする。

4.2.2. 評価結果と考察

2083 行の評価があり、平均値はそれぞれカテゴリタイプでは5点中3.10点、ソングタイプでは5点中3.59点、モデルタイプでは5点中3.87点になった。それぞれの生成タイプについての詳細と考察を述べたいと思う。

4.2.2.1. カテゴリタイプの評価

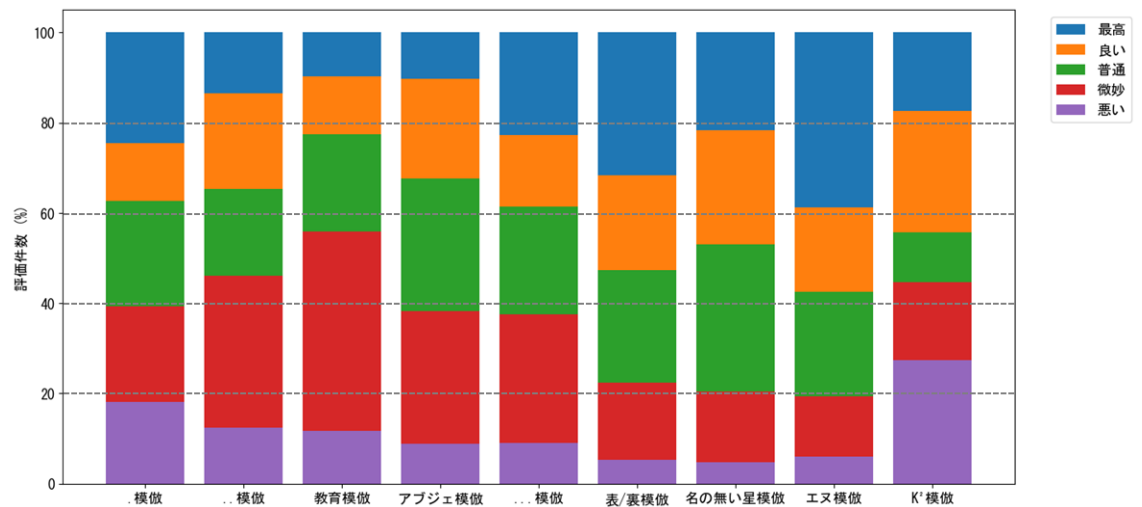
カテゴリタイプでは私が実装した生成タイプのうち、平均的に一番低い点数になってしまった。ここで各カテゴリの平均点や評価の割合に注目するとグラフ1、グラフ2のような結果が得られた。



グラフ1 各カテゴリの平均点

※小数第三位を四捨五入している

※可読性を高める為、「名の無い星が空に堕ちたら模倣」は「名の無い星模倣」と略している。



グラフ 2 各カテゴリの各評価の割合

ここで注目してほしいことは、はじめに述べた特徴 1 の特徴がより強い曲、つまり支離滅裂な曲のカテゴリほど高い点数を得ているところだ。逆に言うと「教育模倣」や「K²模倣」といったカテゴリは原曲の歌詞にストーリー性があるので、それを加味していないカテゴリタイプでは界限の人から高い評価を得にくかったと考察できる。

ただし例外的に「名の無い星が空に堕ちたら模倣」の曲は比較的ストーリー性があるにも関わらず、3.43 点とカテゴリタイプ全体の平均以上の点数を得ることができた。具体的に「最高」と評価された行は以下のような歌詞だ。

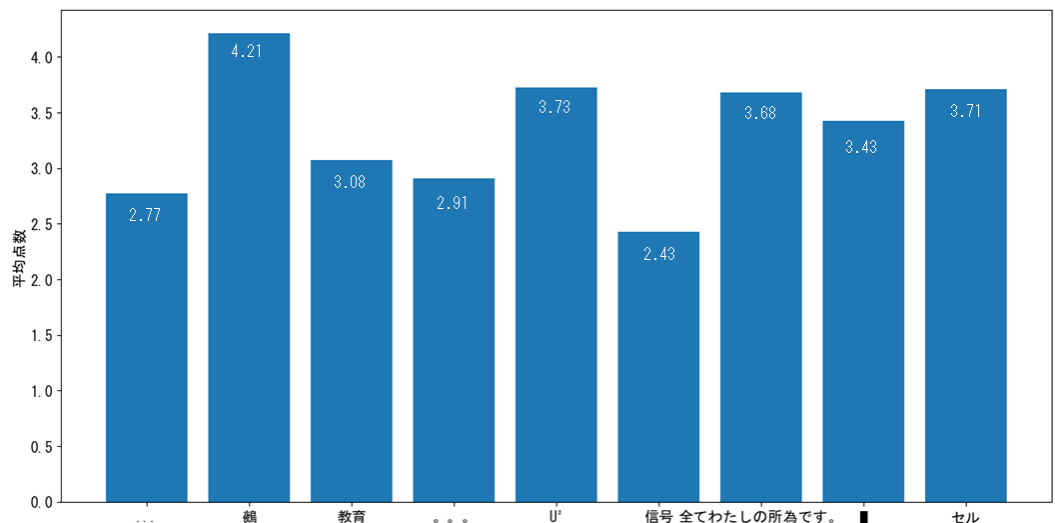
目を覚えたら空が溺れていた
影の陰が伸ばせるでしょう
空の空が揺らめくでしょう
歌を墜ちるわたしのくじら
この日は早い鯨の日で
伸ばせる風の唄は
そっと聞えて思い出したけど
笑い声を墜ちたら空がしまってた響いてた
歌声の遅い鯨が唄に聞えたら
夕やけの美声はとても青くて

夕やけの陰があるでしょう
 手遊びが笑えた
 笑い声のやけはとても遠くって
 星の方へ
 なずむのに
 事を憶え
 わらべの寄り道は
 いうやおぼろ月夜走り抜けてやってた

『名の無い星が空に堕ちたら』やその模倣曲はストーリー性があり、かつ他の原曲とは異なり「夕焼け」、「鯨の歌」、「瑠璃色」といったエモーショナルな単語が登場する。模倣単語にもそういった単語が現れた結果、エモーショナルな歌詞を生成した為、評価が高くなったと考察できる。

4.2.2.2. ソングタイプの評価

ソングタイプは生成する歌詞の原曲を全てあなたの所為です。以外に選べることができ、評価では原曲を含めた計 14 曲の評価をもらった。本研究ではそのうち価件数が 10 件未満でかつ、平均的が 5 点満点では無い曲である 9 曲の評価の分析を行った。各行の点数の平均点はグラフ 3 の通りだ。



グラフ 3 各曲の平均点

結果、カテゴリタイプと同様に支離滅裂な曲のカテゴリほど高い点数を得ていることが考察できた。一方、類似度と平均点の相関は-0.16 で、ほとんど相関は無かった。また、ソングタイプはカテゴリタイプと違い、原曲となる界限曲と類似度をユーザー側が選択できるが故に生成パターンが界限曲の数である約 2000 曲×類似度 1~5 で約 10000 パターンあり、実際に評価が各パターンに分散してしまった為、統計的に有意な結果を得られてるとは言えないだろう。よってこの生成カテゴリの評価を行う為にはさらなる評価件数が必要であると言える。

4.2.2.3. カテゴリタイプとモデルタイプの評価について

さて、カテゴリタイプとモデルタイプの「最高」の評価について、一つ興味深い評価が得られた。

七なる齒に 重ならな庇で 軋轢と重なってた腐らし明瞭な愁い

これらの行の歌詞は文法的に間違っているにも関わらず、「最高」の評価を貰った。この要因として、界限曲には稀に歌詞の流れの響きを変える為に、日本語の文法を崩すことがある。例えば全てあなたの所以です。の『偶』の一節「予測されるは不変のアルファ・アルゴリズム。」 [23] の場合は「予測される」の後に名詞が続いていないので、文法的な間違いがある。界限の人はそういった文法的なミスも加味して「最高」の評価をしたのではないかと考察できる。

4.2.2.4. モデルタイプの評価

モデルタイプは全て歌詞の所為です。上に登録された全ての界限曲を元に gpt2-japanese という高精度な言語モデルを用いて歌詞生成した為、その生成タイプよりも高い 3.87 点もの点数を得ることができた。

「最高」で評価された行の歌詞を詳しく見ると

睡蓮アルウェーンが高圧炉の中、
都合よく書き取る狼煙ですが、
嘘で塗れた水平線の彼方、
警報が発しこちらを向いていました。
希望を引き裂いて悪を為す妖怪たちが、
渴いた音を紡いでいく。
マイクロ波を追いかけて、
誰が為に道を塞ぐのか。

こういった意味不明な歌詞も「最高」と評価されていて更に

分からなくて退廃に沈んだ儘ただの愛よりも、
愛おしそうに抱く。
華奢な骸の春が甘い水を、
夢を見る其の時まで。
わらべ歌の意味は歌詞も消えていた、
心地よい音へと誘うピアノの様。
それは雪の積もるお伽噺で、
蘇る事の無い景色の夢語り。

こういったエモーショナルな歌詞も「最高」と評価されていた。

支離滅裂な歌詞もエモーショナルな歌詞も生成できるモデルタイプは結果的に高い点数を得ることができたのではないだろうか。

5. 結論

5.1. まとめ

本研究では全てあなたの所為です。の模倣曲を作詞できるようなウェブアプリケーション「全て歌詞の所為です。」を制作した。サイト上には「カテゴリタイプ」、「ソングタイプ」、「モデルタイプ」の計 3 つの生成タイプで歌詞を生成できる機能があり、生成した歌詞の各行に対して「最高」を 5 点、「良い」を 4 点、「普

通」を3点、「微妙」を2点、「悪い」を1点で評価を行ったところ、平均値はそれぞれカテゴリタイプでは5点中3.10点、ソングタイプでは5点中3.59点、モデルタイプでは5点中3.87点でモデルタイプの一番点数が高かった。

具体的に、カテゴリタイプの中でも原曲が支離滅裂な歌詞のカテゴリは平均点数が高くなり、逆にストーリー性がある歌詞の原曲のカテゴリは一部例外を除いて平均点が低くなる傾向があった。また、モデルタイプはその両方を考慮して歌詞を生成することができるので高い点数を得ることができた。ソングタイプは統計的な有意な結果を得ることができなかったが、概ねカテゴリタイプと同じように支離滅裂なほど点数が高くなる傾向があった。また、カテゴリタイプとソングタイプに関しては文法的な間違いのある歌詞にも関わらず、最も良い「最高」の評価をしているケースがあった。

5.2. 今後の課題

今後の研究としては歌詞生成時に文法的に間違えている歌詞を修正した場合の各生成タイプの評価の影響について調査したいと考えている。また、gpt2-generate.py のオプションである top_k や temperature の値を変えた場合、より界限曲に近い文章は生成できるか研究を行いたい。

参考・引用・参考文献

- [1] 全てあなたの所為です。(2020) 『エヌ』
- [2] 全てあなたの所為です。(2022) 『K²』
- [3] 全てあなたの所為です。(2020) 『...』
- [4] 全て夢見の為す事です。(2022) 『|||』
- [5] Abe, C., & Ito, A. (2012, December). A Japanese lyrics writing support system for amateur songwriters. In Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (pp. 1-4). IEEE.

- [6] Rodrigues, M. A., Oliveira, A., Moreira, A., & Possi, M. (2022, May). Lyrics Generation supported by Pre-trained Models. In The International FLAIRS Conference Proceedings (Vol. 35).
- [7] Chen, Y., & Lerch, A. (2020, December). Melody-conditioned lyrics generation with seqgans. In 2020 IEEE International Symposium on Multimedia (ISM) (pp. 189-196). IEEE.
- [8] Takahashi, R., Nose, T., Chiba, Y., & Ito, A. (2020, October). Successive Japanese Lyrics Generation Based on Encoder-Decoder Model. In 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE) (pp. 126-127). IEEE.
- [9] Riedl, M. (2020). Weird AI Yankovic: Generating Parody Lyrics. arXiv preprint arXiv:2009.12240.
- [10] Mocobeta. (2022, February 25). WELCOME TO JANOME'S DOCUMENTATION! (JAPANESE). Janome v0.4 Documentation (Ja). <https://mocobeta.github.io/janome/>
- [11] Tedboy. (2016, September 13). API — Gensim. NLP APIs. https://tedboy.github.io/nlps/api_gensim.html
- [12] Tanreinama. (2022, September 11). Tanreinama/Gpt2-Japanese: Japanese GPT2 Generation Model. GitHub. <https://github.com/tanreinama/gpt2-japanese>
- [13] 全て幻だろうか。(2020)「◆[short]」
- [14] 全てあなたの音です。(2022)「♪♪」
- [15] 何時か忘れて仕舞うのでしょうか。(2022)「Morgenrot」
- [16] 全て行方の所為です。(2022)「𐄂」
- [17] 全て幻だろうか。(2020)「◇[short]」

[18] 全てあの世の所為です。(2022)「E403」

[19] 全てあなたの所為です。(2018)「.」

[20] 全てあなたの所為です。(2018)「..」

[21] 全てあなたの所為です。(2018)「教育」

[22] 全てあなたの所為です。(2018)「アブジェ」

[23] 全てあなたの所以です。(2021)「偶」