2025 年度 iTL 先端的プロジェクト
奨学金最終報告書

日本語リポグラム自動生成のための自然言語処理技術の開発

中央大学 国際情報学部
22G1104002B 橋本葵
指導教員：飯尾淳
提出日：2026 年 3 月 6 日

目次

第1章　はじめに

本章では、日本語におけるリポグラム自動生成という課題の背景と問題設定を整理し、本研究が提案する「日本語リポグラム生成の枠組み」の目的と位置づけを明確にする。とくに、文字・音レベルの負制約の下で、制約遵守（禁止文字の完全排除）と自然さ・意味保持を両立することが難しい点を述べ、その難しさを扱うための設計単位として枠組み（フレームワーク）を提示する。

1.1　研究背景

本節では、リポグラムの概要とその背景を整理し、制約付き文章生成の文脈における位置づけを述べる。とくに、日本語では表記と読みが必ずしも一致しないため、禁止条件をどの水準（表記／読み）で課すかがタスク定義に直結する。この点を踏まえ、本研究が読みベース制約を扱う意義を明確にする。

1.1.1　リポグラムの定義

リポグラム（lipogram）とは、特定の文字を意図的に用いないという制約の下で書かれた文章・作品を指す。単なる言語遊戯としての側面に加え、「制約を満たしながら意味の通る文章を作る」という点で、制約付き文章生成（constrained text generation）の具体例として捉えることができる。

近年は、敬語変換・スタイル変換・NG ワード回避など、条件を満たす出力を生成するタスクが増えており、リポグラムはその中でも「禁止条件がきわめて細かい（文字／音）」という特徴を持つ。

1.1.2　歴史的事例

リポグラムには著名な歴史的事例が存在する。たとえば Perec（1969）の『La Disparition』は、特定文字を用いないという制約の下で長編小説を成立させた代表例として知られている。このような事例は文化的関心の対象である一方、工学的には「ごく単純な条件（ある文字を出さない）」であっても、人間にとっては創作上の負担が大きく、機械にとっては制御が難しいことを示す例でもある。すなわち、リポグラムは出力制御の難しさを可視化するテストベッドとして機能する。

1.1.3　制約付き文章生成との関連

制約付き文章生成の文脈では、「望ましくない語や表現を出力に含めない」「所望のスタイルに合わせる」といった制約が扱われる。とくに実運用の対話システムや Web サービスでは、有害表現や不適切表現を含まない出力が求められるため、出力制御は重要である。し

かし、生成モデルに対してプロンプトで禁止条件を与えただけでは、禁止語の混入を完全に防げないことが多い。出力後にフィルタリングや単純置換を行う方法もあるが、文法性の破綻や意味の欠落が生じやすい。

リポグラムの特徴は、制約の粒度が文字・音レベルであり、違反が「1 文字でも混入すれば失敗」として扱われる点にある。NG ワード回避のように「多少の揺れを許容できる」場面と異なり、リポグラムでは完全排除（ゼロ違反）が本質的要件となる。この厳格さが、生成過程の制御をより難しくする。

1.1.4　日本語におけるリポグラムの難しさ

日本語でリポグラムを扱う場合、アルファベット言語とは異なる難しさが現れる。

第一に、日本語は漢字・ひらがな・カタカナが混在し、同じ読みを複数の表記で書ける。たとえば、表記上は禁止文字を含まなくても、読みとしては禁止音を含む場合がありうる。そのため、禁止条件を「表記（文字列）」に課すのか「読み（仮名列）」に課すのかによって、タスクの難度や生成可能な表現の範囲が大きく変化する。

第二に、助詞や活用語尾など、高頻度に現れる要素に禁止文字が含まれると、文全体の構造を変えなければ制約を満たせないことがある。単語の置換だけでは回避できず、文の言い回しや構文を再設計する必要が生じる。

第三に、日本語は空白による単語区切りが明示されないため、文字列レベルの機械的処理では、意味を保った言い換えの単位を定めにくい。以上の理由から、日本語リポグラムは人手でも難度が高く、自動生成はさらに難しい課題となる。

1.2　問題設定

本節では、本研究で扱うタスクを「禁止文字を一切含まない文章生成」として整理し、なぜ単純な後処理（削除・置換）では十分でないかを述べる。あわせて、制約遵守、意味保持、自然さという複数要件の同時達成が困難であることを示し、評価と再生成制御が必要となる理由を明確にする。

1.2.1　禁止文字を含まない文章生成という課題

本研究で扱う中心課題は、「与えられた日本語文と禁止文字集合に対して、禁止文字を一切含まない日本語文を生成する」ことである。単純なアプローチとしては、禁止文字を含む箇所を削除・置換する方法が考えられる。しかし、禁止文字が助詞や語尾に含まれる場合、それを機械的に削除すると文法性が崩れやすい。また、置換で回避しようとしても、置換候補の読みが禁止文字に引っかかるなど、局所修正だけでは解決できないケースが多い。

したがって、この課題は「禁止文字を除去する」だけではなく、必要に応じて文全体を再構成しながら制約を満たす生成を行う問題として捉える必要がある。

1.2.2　文法性・意味保持・自然さの両立の困難性

制約付き生成では、一般に「制約遵守」と「品質（意味保持・自然さ）」の間にトレードオフが生じる。禁止文字を厳格に避けようとするほど、言い換えの自由度が制限され、不自然な表現や意味の逸脱が起こりやすい。一方、意味や自然さを優先すると、禁止文字が混入するリスクが高まる。とくにリポグラムでは、制約違反が 1 文字でも許されないため、このトレードオフは顕著になる。

さらに、日本語リポグラムでは「表記上の制約」と「読み上の制約」がずれる可能性があるため、品質と制約の両方を評価するための枠組み（指標の定義、測定手順）が必要になる。本研究では、制約遵守率（Constraint Satisfaction Rate）を主指標としつつ、意味保持や自然さを測る指標を併用して多面的に評価する。

1.2.3　日本語特有の制約問題

日本語では、活用語尾や助詞が高頻度で現れるため、禁止文字がそこに含まれると回避が難しい。また、読みベースの制約を採用する場合、漢字語でも読みが禁止音に引っかかることがあり、表記の工夫だけで回避できない。さらに、形態素解析に依存して読みを得る以上、未知語や解析誤りが存在しうる。これらの要因は、「読みを取れば制御できる」という直観を成立させる一方で、制御の前提（読みの取得と正規化、失敗時の扱い）を明示的に定義する必要があることも示している。

1.3　研究目的

本節では、前節で述べた課題に対して、本研究が何を提案し、何を明らかにすることを目標とするかを述べる。とくに、日本語リポグラム生成を実装・評価可能な単位に分解した枠組みを提示し、その有効性を検証するという立場を明確にする。

本研究の目的は、日本語リポグラム生成を、実装・評価可能な形で整理した「枠組み（フレームワーク）」として提案し、その枠組みが制約遵守と品質の両立にどのように寄与するかを明らかにすることである。具体的には、入力仕様（入力文と禁止集合）、制約検査（表記／読み）、生成（LLM による候補生成）、再生成制御（失敗時の再試行・局所修正）を明確に分離し、それらを接続するパイプラインとして定式化する。

この枠組みの中で、本研究は二つの生成方式を取り扱う。第一は、入力全体を一度に書き換える一発生成方式（oneshot）であり、プロンプトのみで制約を伝える素朴なベースラインとして位置づける。第二は、禁止文字を含む箇所を検出し、その箇所に対して逐次的に介入して書き換える逐次生成方式（sequential）である。方式比較は、本研究の主張を「方式の優劣」そのものに置くのではなく、枠組みの要素（たとえば読みベース制約チェックや局所介入、反復的な再生成制御）が実際に効果を持つことを検証するための手段として行う。

加えて、本研究は生成品質を「意味保持」と「自然さ（文法性・流暢性）」の観点で評価

する。制約遵守だけを満たす出力は、実用上・創作支援上は価値が限定的であるため、どの程度まで元文の意味を維持できているか、どの程度自然な日本語として読めるかを併せて検討する。評価方法の詳細（指標の定義と測定手順）は第 3 章および第 6 章で述べる。

1.4　研究の意義

本節では、本研究が扱う日本語リポグラム生成を、学術的・応用的観点からどのように位置づけるかを述べる。リポグラムそのものの面白さに留めず、出力制御や言い換え支援の基盤技術としての意義を整理する。

1.4.1　文学的創作支援

リポグラムは、制約下での創作を促す技法として知られる。本研究で提案する枠組みが実用化されれば、創作者が「禁止文字（禁止音）」を指定し、複数の候補表現を得ながら推敲する、といった創作プロセスを支援できる可能性がある。とくに、日本語では制約を満たす表現探索が難しいため、自動生成による候補提示は負担軽減につながりうる。

1.4.2　言語教育支援

制約付き作文は、語彙選択や文法構造への意識を高める練習として利用できる。たとえば、特定の仮名を使わずに同じ意味を表す言い換えを考える過程は、同義表現や言い回しの学習に資する。本研究の枠組みは、禁止条件を明確に与え、生成候補を比較できるため、教材作成や学習支援への応用可能性がある。

1.4.3　コミュニケーション支援

読み（音）に対する制約は、発音しづらい音を避けたい状況や、特定音を含まない代替表現が必要な場面と親和性がある。たとえば、発話支援・入力支援の文脈では、「言いにくい音を避けた言い換え」を候補として提示することが有用になりうる。本研究は直接の支援システム構築を目的とはしないが、読みベース制約を中核に据えることで、そのような応用への足がかりとなる設計知見を提供できる。

1.5　本研究の貢献

本研究は、タスク定義、実装枠組み、評価プロトコルを一体として整備し、日本語リポグラム生成を体系的に扱う基盤を提供することを目指す。以下に、本研究の主な貢献をまとめる。

日本語リポグラム生成を「入力仕様→制約検査（表記／読み）→生成→再生成制御」に分解した枠組みとして定式化し、同一枠組み内で生成方法を切り替え可能な設計として示す。

読みベース制約チェック（形態素解析で得た読みの正規化を含む）を中核要素として組み込み、漢字・かなの表記差をまたいだ禁止音の検出・制御を可能にする。

評価用データセットと実験パイプライン（自動指標・ログ収集）を整備し、ベースライン／アブレーションを含む比較実験により枠組みの妥当性とトレードオフを示す。

生成結果の比較や失敗例収集を支援する Web アプリケーションを実装し、分析および主観評価の基盤として利用可能にする。

第2章　関連研究

本章では、本研究が位置づけられる制約付き文章生成（constrained text generation）の研究動向を整理した上で、リポグラムに関する先行事例と、日本語における制約生成の難しさを概観する。とくに、本研究が扱う「禁止条件を文字・読み（音）レベルで厳格に課し、違反をゼロにする」設定は、NG ワード回避やスタイル変換と近い構造を持ちながらも、制約の粒度と厳格さの点で異なる。これらの差分を踏まえ、本研究が提案する枠組み（入力仕様、制約検査、生成、反復制御）の必要性と位置づけを明確にする。

2.1　制約付き文章生成

本節では、制約付き文章生成に関する代表的な課題として、NG ワード回避やスタイル変換を取り上げ、本研究が扱う「禁止条件の完全遵守（ゼロ違反）」という設定との共通点と相違点を整理する。あわせて、LLM による生成を前提とした場合に、プロンプトのみの制御では十分でないことを述べ、本研究が採用する「検査と再生成を組み合わせる」方針へ接続する。

制約付き文章生成の文脈では、「望ましくない語や表現を出力に含めない」「所望のスタイルに合わせる」といった制約が扱われる。とくに実運用の対話システムや Web サービスでは、有害表現や不適切表現を含まない出力が求められるため、出力制御は重要である。しかし、生成モデルに対してプロンプトで禁止条件を与えただけでは、禁止語の混入を完全に防げないことが多い。出力後にフィルタリングや単純置換を行う方法もあるが、文法性の破綻や意味の欠落が生じやすい。

リポグラムの特徴は、制約の粒度が文字・音レベルであり、違反が「1 文字でも混入すれば失敗」として扱われる点にある。NG ワード回避のように「多少の揺れを許容できる」場面と異なり、リポグラムでは完全排除（ゼロ違反）が本質的要件となる。この厳格さが、生成過程の制御をより難しくする。関連する試みとして、負の語彙制約（negative lexical constraints）を用いたデコーディングにより、必要箇所を確実に書き換える枠組みも提案されている（Kajiwara, 2019）。

2.1.1　NG ワード除去研究

制約付き文章生成の代表例として、NG ワード除去や有害表現回避が挙げられる。これらは、出力から特定語を排除するという点でリポグラムと同型の「負制約」を持つ。一方で、多くの実運用では禁止語は単語・フレーズ単位で管理されることが多く、単純な置換やマスク、あるいは生成後のフィルタリングでも一定の効果が得られる場合がある。

しかし、言語モデルの生成においては、単純に「bad words」を禁止するような手法だけでは望ましくない出力を十分に抑えきれないことも報告されている（Gehman ら, 2020）。

しかし、禁止語の検出・削除を出力後処理として行うだけでは、文法性の破綻や意味欠落が生じやすい。禁止要素が文の骨格（主語、述語、助詞、否定など）に関わる場合はなおさらであり、局所的な削除ではなく文全体の再構成が必要となる。このため、近年のアプローチでは「検査と再生成を組み合わせる」枠組みが重要になる。具体的には、次のような流れで制御する。

- 制約違反の検出
- 違反箇所の再生成または編集
- 再検査と反復

本研究が扱う日本語リポグラムは、禁止条件が文字・音レベルにまで細分化されるため、この枠組みの必要性がより強く現れる。

生成過程での制約制御としては、負の語彙制約（negative lexical constraints）をデコーディングに組み込む枠組みも提案されている（Kajiwara, 2019）。

2.1.2　表現変換（敬語・ポジティブ表現化）

スタイル変換（敬語変換、ポジティブ表現化、感情・文体制御など）は、入力文を意味的に保ちながら別の条件を満たす文へ変換するタスクとして研究されてきた。スタイル変換では、意味保持とスタイル達成のトレードオフをどのように制御するかが中心課題となり、評価においても自動指標と人手評価の併用、あるいは複数指標の組合せが議論されている。

リポグラム生成も、「意味を保ちつつ、所望の条件を満たす」という点でスタイル変換と共通する構造を持つ。ただし、スタイル変換では「文体が一定程度変化していればよい」などの連続的な目標として扱えるのに対し、リポグラムでは禁止文字の混入が 1 文字でもあれば失敗となる、離散的かつ厳格な条件となる点が異なる。したがって、本研究ではスタイル変換の評価設計の知見を参照しつつも、制約遵守を主指標として明確に位置づけ、意味保持・自然さとの関係を多面的に捉える必要がある。

非並列データからのスタイル変換としては、潜在表現のアラインメントに基づく手法（Shen ら, 2017）や、Delete–Retrieve–Generate 型の枠組み（Li ら, 2018）が代表例として挙げられる。

意味保持の自動評価としては、文埋め込みに基づく類似度のような指標が広く用いられており、代表例として BERT（Devlin ら, 2019）が挙げられる。

2.1.3　LLM を用いた制約生成

前節までで述べた通り、プロンプトのみで禁止条件を完全に守らせることは難しい。この課題に対し、近年は制約付きデコーディングや negative constraints（負の語彙制約）をデコーディング過程に組み込む研究が報告されている。ただし、商用 API として提供される LLM では、内部のデコーディング制御を利用者側で細かく指定できない場合が多い。そのため、

本研究では API ベースの LLM を前提とし、次を組み合わせることで、実用的に制約遵守を高めるアプローチをとる。

- 生成後の制約検査
- 再生成・局所修正の反復

2.2 リポグラム生成研究

本節では、リポグラムに関する先行事例と公開実装を概観し、とくにアルファベット言語を前提とした手法が日本語へ直接適用しにくい理由を整理する。これにより、日本語では表記と読みのずれを踏まえた制約定義と制御が必要であることを明確にする。

2.2.1 英語・フランス語研究

リポグラムの先行事例は、英語やフランス語などアルファベット言語に関するものが中心である。たとえば Perec（1969）の「La Disparition」は、特定文字を用いない長編作品として著名であり、制約下での創作が成立することを示してきた。アルファベット言語では、文字と音の対応が日本語に比べて単純であることが多く、「特定文字を含まない」という条件が比較的直接的に扱える。この点は、工学的にも、文字列レベルの検査・探索と相性がよい。

一方、日本語では表記体系が複雑であり、同じ音を複数の表記で書ける。したがって、「表記上の禁止」と「読み上の禁止」は必ずしも一致せず、どちらを制約として採用するかがタスク定義に直結する。本研究は、この差分が顕在化する日本語を対象として、読みベース制約を含む定義と評価を行う点に特徴がある。

2.2.2 GitHub 実装例

公開実装としては、特定文字を含む単語を辞書的に置換する方式、あるいは使用可能な語彙を列挙して探索する方式などが見られる。アルファベット言語では、語彙が空白で区切られるため、単語単位の置換や探索が比較的自然に定義でき、文字列検査も単純である。

しかし、日本語では空白による単語区切りが明示されず、表層の文字列操作だけでは「意味を保った置換」を単語単位で行うことが難しい。また、漢字表記は表層文字列に禁止仮名が現れない一方で、読みとしては禁止音を含みうるため、表層文字列だけを見た検査では不十分になりうる。以上から、英語前提の単語置換ベース実装をそのまま日本語へ適用することは困難であり、日本語に合わせた制約検査と再生成制御が必要となる。

関連する公開実装の例として、フランス語リポグラム生成を扱う GitHub リポジトリも報告されている（Loujain, 2025）。

2.2.3 既存手法の限界

既存のリポグラム生成（とくに簡易実装）では、制約違反を避けること自体は達成できて

も、意味保持や文法性（自然さ）を十分に担保できない場合がある。また、制約遵守が主目的であるがゆえに、意味保持や自然性をどのように測り、どの程度を許容するかといった評価基準が明確でないことも多い。

本研究は、制約遵守（ゼロ違反）を主指標として明確化しつつ、意味保持・自然性・計算コストを併せて測定することで、「制約を守れるが不自然」や「自然だが制約違反」といったトレードオフを定量的に議論できる枠組みを整備する点に意義がある。

2.3 日本語の制約生成

本節では、日本語の言語的特性が制約付き生成に与える影響を整理し、本研究が読みベース制約を重視する理由を述べる。あわせて、NG ワード回避との違い（制約単位と要求精度）を明確化し、日本語リポグラム生成が独自の難しさを持つことを示す。

2.3.1 日本語リポグラム研究の未整備

日本語リポグラムを主題とした体系的研究は多くない。個人レベルの創作や小規模な試行は散発的に存在するものの、タスク定義、評価指標、実験プロトコルを揃えた比較研究は限定的である。この状況は、本研究にとって「関連研究が薄い」ことを意味するだけでなく、逆に、タスク定義と評価枠組みを明示すること自体が貢献になりうることを示している。

2.3.2 日本語特性（形態素・表記揺れ・助詞）

日本語は、次の性質を持つ。

- 形態素境界が空白として現れにくい
- 漢字・ひらがな・カタカナが混在し表記揺れが大きい
- 助詞や活用語尾など高頻度要素が文の骨格に強く関与する

これらは、禁止条件が一部の高頻度仮名を含む場合に、局所的な置換では回避できず、文の構造を変える必要が生じることを意味する。また、読みベース制約を課す場合は、形態素解析により読みを得て検査することが実装上の前提となるため、解析失敗や未知語の扱いが制御の一部となる。

2.3.3 NG ワード研究との差分

NG ワード回避と日本語リポグラムは、いずれも「望ましくない要素を含まない出力」を求める点で共通するが、要求精度と制約単位が異なる。リポグラムでは、文字・音レベルの禁止であり、違反が 1 文字でも許されないため、誤検知・漏れの許容度が低い。また、読みベース制約では、表記上の文字列検査だけでなく、読みへの写像（形態素解析と正規化）を含む判定が必要となる。したがって、本研究では、制約検査と再生成制御を明確に分離した枠組みとして設計し、評価においても「読みベースで違反ゼロ」を主指標として扱う。

2.4　本研究の位置づけ

以上の整理を踏まえると、本研究は、日本語リポグラム生成という未整備領域に対して、読みベース制約を含む明確なタスク定義と、実装・評価可能な枠組みを提示する応用研究として位置づけられる。具体的には、入力仕様（禁止集合の正規化・展開）、制約検査（表記／読み）、生成（LLM による候補生成）、再生成制御（反復と停止条件）を分離し、パイプラインとして統合する。また、同一枠組み内で生成内容を切り替え可能にすることで、「逐次的介入」や「反復制御」といった要素の寄与を、ベースラインおよびアブレーションを通じて検証できる形にする。さらに、生成結果の比較や失敗例収集を支援する Web アプリケーションを併設することで、定量評価に加え定性分析や主観評価へ発展可能な基盤を提供する。

第3章　問題定義

本章では、本研究で扱う日本語リポグラム生成タスクを形式的に定義する。とくに、入力（元文と禁止集合）、出力（生成文）、および制約遵守の判定方法（表記ベース／読みベース）を明確化し、以降の手法設計（第 4 章）と実験評価（第 6〜7 章）が再現可能になるように整理する。また、本研究は「読み（仮名列）を取って制御する」という立場を取るため、形態素解析による読み取得と正規化、および解析失敗時の扱いも含めて前提を定める。

3.1　タスク定義

本節では、本研究の入力・出力・成功条件を定義する。とくに、禁止集合の扱い（行指定を含む入力Bの正規化）と、表記ベース制約／読みベース制約の二つの制約定義を整理する。

3.1.1　入力の定義

本研究の入力は、日本語文xと禁止集合Bである。ここでxは、ことわざ・例文などの文単位テキストを想定する。禁止集合Bは、ひらがな 1 文字の集合として指定する場合（例:「い、さ」）に加え、「あ行」のように行単位で複数の仮名をまとめて禁止する指定も許す。

以降、行指定などを含むBを、最終的にひらがな 1 文字集合へ展開・正規化した集合をB^*と表す。たとえば「あ行」を指定した場合は、$B^* = \{$あ、い、う、え、お$\}$となる。複数指定がある場合は和集合として扱う。

3.1.2　出力の定義

出力は、日本語文yである。yは、指定された禁止集合に関する制約を満たすことに加えて、日本語として自然に読めること、および元文xの主要な意味内容を大きく損なわないことが望ましい。本研究では、制約遵守を必須要件（違反は失敗）として扱い、意味保持・自然性は品質要件として評価指標により測定する。制約遵守には二つの定義を用意し、目的に応じて使い分ける。

- 表記ベース制約：出力文yの表層文字列に禁止集合B^*が一切含まれない。
- 読みベース制約：出力文yを形態素解析して得られる読みの仮名列に、禁止集合B^*が一切含まれない。

以降の実験では、漢字表記の違いをまたいだ禁止音の回避を扱うため、読みベース制約を主制約として採用する。

3.1.3　文法性・意味保持要件

文法性・自然性とは、出力文yが日本語として自然に読めることを指す。多少の文体変化

（丁寧体から常体への変化など）は許容しうるが、意味解釈が困難になるほどの語順・活用の破綻は望ましくない。

意味保持とは、出力文yが元文xの主要な意味内容を保っていることを指す。完全な同義を要求するのではなく、「言い換えとして許容できる範囲で意味が対応している」ことを目標とする。意味保持と自然性の評価方法（自動指標・主観評価）は第 6 章で述べる。

3.1.4　前提とスコープ

本研究は、禁止集合を「ひらがな」により与える。句読点・記号・英数字などは、禁止集合の対象外とする。また、本研究の主タスクは文単位のリライト／生成とし、段落や長文への適用は今後課題として扱う。

読みベース制約は、形態素解析器が返す各トークンの読み（多くの場合カタカナ）を、ひらがなへ正規化した仮名列に対して課す。ここで、ひらがなへの正規化とは、カタカナを対応するひらがなへ変換する処理を指す。読みを連結して得られる仮名列を Read(y) とする。

形態素解析では未知語・解析失敗が起こりうるため、本研究では安全側の扱いとして「読みが取得できないトークンが存在する場合、その出力は読みベース制約の判定に失敗（違反）とみなし、再生成・再書き換えの対象とする」と定める。

3.2　制約の種類

本節では、本研究で用いる制約の種類を整理する。表記ベース制約と読みベース制約を定義した上で、実験で扱う制約強度（単音禁止／行禁止）を位置づける。

3.2.1　文字ベース制約

表記ベース制約は、出力文yの表層文字列に禁止集合B^*が一切含まれないこととして定義する。ここで、禁止集合がひらがなで与えられることを踏まえ、表層文字列は必要に応じて正規化（例：カタカナをひらがなへ変換）した上で検査する。正規化後の表層文字列を$\mathrm{Surf}(y)$とし、$\mathrm{Surf}(y)$にB^*の要素が含まれないことを表記ベース制約の成功条件とする。

3.2.2　読みベース制約

読みベース制約は、$\mathrm{Read}(y)$に禁止集合B^*が一切含まれないこととして定義する。具体的な判定手順は次の通りである。

- 出力文yを形態素解析し、トークン列を得る。
- 各トークンに対して読みを取得し、ひらがなへ正規化する。
- 正規化後の読みを連結して$\mathrm{Read}(y)$を得る。
- $\mathrm{Read}(y)$にB^*の要素が含まれるかを検査する。

日本語では同一表記に複数の読みがありうるが、本研究では形態素解析器が返す読みを

採用し、その出力に従って判定する。読みが取得できない場合は前節の規定に従い失敗とみなす。

3.2.3　制約強度

本研究では、禁止集合の大きさや、禁止される仮名の頻度に応じて制約の強度が変化すると考える。実験では、単音禁止（例：ひらがな 1 文字の禁止）を比較的弱い制約、行禁止（例：「あ行」の禁止）をより強い制約として扱い、これらを条件として比較する。

3.3　評価基準（指標の定義）

本節では、本研究で用いる評価指標を定義する。制約遵守率（Constraint Satisfaction Rate）を主指標として位置づけ、意味保持・自然性・書き換えの性質・計算コストを補助指標として扱う。

本研究は、制約遵守（違反ゼロ）を最優先の要件として扱い、制約遵守率を主指標とする。制約遵守率は、評価対象の入力集合に対して、制約を満たす出力が得られた割合として定義する。読みベース制約を主制約とする場合は、Read(y)に禁止集合が含まれない出力の割合を指す。

補助指標として、次の観点を用い、主指標とのトレードオフを議論する。各指標の具体的な計算方法と測定手順は第 6 章で述べる。

1. 意味保持
2. 法性・自然性
3. 書き換えの性質（語彙置換率など）
4. 計算コスト（実行時間や再試行回数）

3.3.1　主指標：制約遵守率

評価対象の入力がN件あり、そのうち読みベース制約のチェックで制約違反が一度も検出されなかった（成功した）出力の件数をN_{success}とすると、制約遵守率は次式で定義される。

$$\text{ConstraintRate} = \frac{N_{\text{success}}}{N} \tag{1}$$

第 6〜7 章では、本指標を方式（oneshot／sequential）および制約タイプ（単音禁止／行禁止など）ごとに集計して比較する。

3.3.2　補助指標：書き換えの性質・計算コスト

本研究では、制約遵守を満たした出力について、書き換えの程度や生成結果の性質を把握するために VRR と TTR を用いる。また、不自然な反復の検出や実用性の観点から n-gram 反復率と実行時間を補助指標として用いる。

VRR は、元文と生成文の内容語トークン列を比較し、置き換わった（対応しない）トークンの割合を測定する指標である。元文の内容語トークン列をxとし、対応付けの結果として「置換された」とみなすトークン数を$|x_{\mathrm{df}}|$とすると次式で表す。

$$\mathrm{VRR} = \frac{|X_{\mathrm{diff}}|}{|X|} \tag{2}$$

挿入・削除によりトークン数が一致しない場合があるため、実装では最長共通部分列（Longest Common Subsequence; LCS）に基づく近似により、「順序を保って共通に残ったトークン数」を見積もったうえで頑健に算出する。

TTR は、生成文におけるユニーク語彙の割合を測定し、語彙の偏りや過度な繰り返しがないかを確認する指標である。生成文の内容語トークン列をyそのユニーク語彙集合を$\mathrm{types}(y)$とすると、次式で定義される。

$$\mathrm{TTR} = \frac{|\mathrm{types}(Y)|}{|Y|} \tag{3}$$

n-gram 反復率は、同一の bi-gram や tri-gram がどの程度繰り返されているかを測定し、不自然な反復を検出するために用いる。全 n-gram 数をN_{ngram}、2 回以上出現した n-gram の数を$N_{\mathrm{ngram、rep}}$とすると、以下として表す。

$$\mathrm{RepRate}_n = \frac{N_{\mathrm{ngram、rep}}}{N_{\mathrm{ngram}}} \tag{4}$$

実行時間は、1 入力パターンあたりの処理時間（秒）である。逐次生成方式は局所介入と再試行を含むため、成功率とのトレードオフとして実行時間が増加しうる。実行時間は方式間比較の補助指標として用いる。

3.4　サブタスク

本節では、同一枠組み内で比較する生成方式（oneshot／sequential）と、再生成・局所修正をサブタスクとして整理する。

本研究は、同一の枠組み内で次のサブタスク（生成方式）を扱う。

- 一発生成（oneshot）：元文xと禁止集合B^*をまとめて LLM へ与え、1 回の生成で出力文yを得る。失敗時は基本的に全文の再生成を行う。
- 逐次生成（sequential）：形態素解析により違反箇所を検出し、違反を含むトークン（または局所区間）に対して順次言い換えを行う。各ステップで再検査し、必要に応じて再試行する。
- 再生成・局所修正：制約違反や品質不足が検出された場合に、プロンプトや対象範囲を調整しながら再生成・修正を反復する。

3.5　研究課題

本研究では、以下の研究課題（Research Question; RQ）を設定し、評価指標と実験条件により検証する。

RQ1：日本語リポグラム生成において、逐次生成方式（sequential）は一発生成方式（oneshot）に比べて、どの程度制約遵守率を改善できるか。

RQ2：制約遵守を満たしたうえで、元文の意味保持や自然さをどの程度保てるか。その際、書き換えの性質（例：語彙置換率や語彙多様性）にはどのような差が生じるか。

RQ3：制約タイプ（単音禁止／行禁止など）や文長・文ジャンルの違いによって、両方式の得意・不得意はどのように変化するか。

第4章　手法

本章では、第 3 章で定義した日本語リポグラム生成タスクに対して、本研究が用いる生成枠組み（フレームワーク）を述べる。枠組みは、入力文と禁止集合を受け取り、制約検査（表記／読み）と生成（LLM）と再生成制御を組み合わせて、制約違反のない出力を得るパイプラインとして構成される。本研究では、この枠組みの内部実装として、一発生成方式（oneshot）と逐次生成方式（sequential）を同一条件で比較できるように設計する。

4.1　全体アーキテクチャ

本節では、本研究が採用する日本語リポグラム生成枠組みの全体像を示す。とくに、制約チェックと生成を往復させるパイプライン構造と、oneshot と sequential の二方式がその中でどのように位置づくかを整理する。

4.1.1　システム概要

本研究で用いる生成枠組みは、次のモジュールから構成される。

第一に入力処理モジュールは、入力文xと禁止集合Bを受け取り、第 3 章で定義した正規化・展開により禁止集合B^*を得る。第二に制約検査モジュールは、出力候補yが制約を満たすかを判定する。判定は表記ベース制約（$\mathrm{Surf}(y)$にB^*が含まれない）および読みベース制約$\mathrm{Read}(y)$にB^*が含まれない）に基づく。第三に生成モジュールは、LLM を用いて候補文を生成する。第四に再生成制御モジュールは、制約違反や解析失敗が生じた場合に、生成の対象範囲（局所／全文）や指示（プロンプト）を調整しながら再試行を行い、停止条件に到達するまで反復する。

以降では、この枠組みの処理の流れを CHECK→GENERATE→REWRITE の三段階として説明し、その上で oneshot と sequential の違いを述べる。

4.1.2　パイプライン構造

CHECK 段階では、候補文が禁止集合B^*に違反していないかを検査する。検査は、表記ベース$\mathrm{Surf}(y)$と読みベース$\mathrm{Read}(y)$の双方で行う。sequential 方式では、この段階で違反箇所（違反トークン）を同定し、後続の局所書き換え対象を決定する。

GENERATE 段階では、LLM により新たな候補を生成する。oneshot 方式では、入力文全体を対象に 1 回の生成で候補文yを得る。一方、sequential 方式では、違反箇所を含むトークン（または局所区間）を対象として、文脈を保ちながら置換候補を生成する。

REWRITE 段階では、生成された候補を適用して文を更新し、再度 CHECK 段階で制約を満たすかを判定する。制約違反が残る場合は、再生成制御により、プロンプトの変更、対象範囲の拡大（局所→全文）、再試行回数の上限などの規約に従って反復する。

図 1 に、日本語リポグラム生成パイプラインを示す。

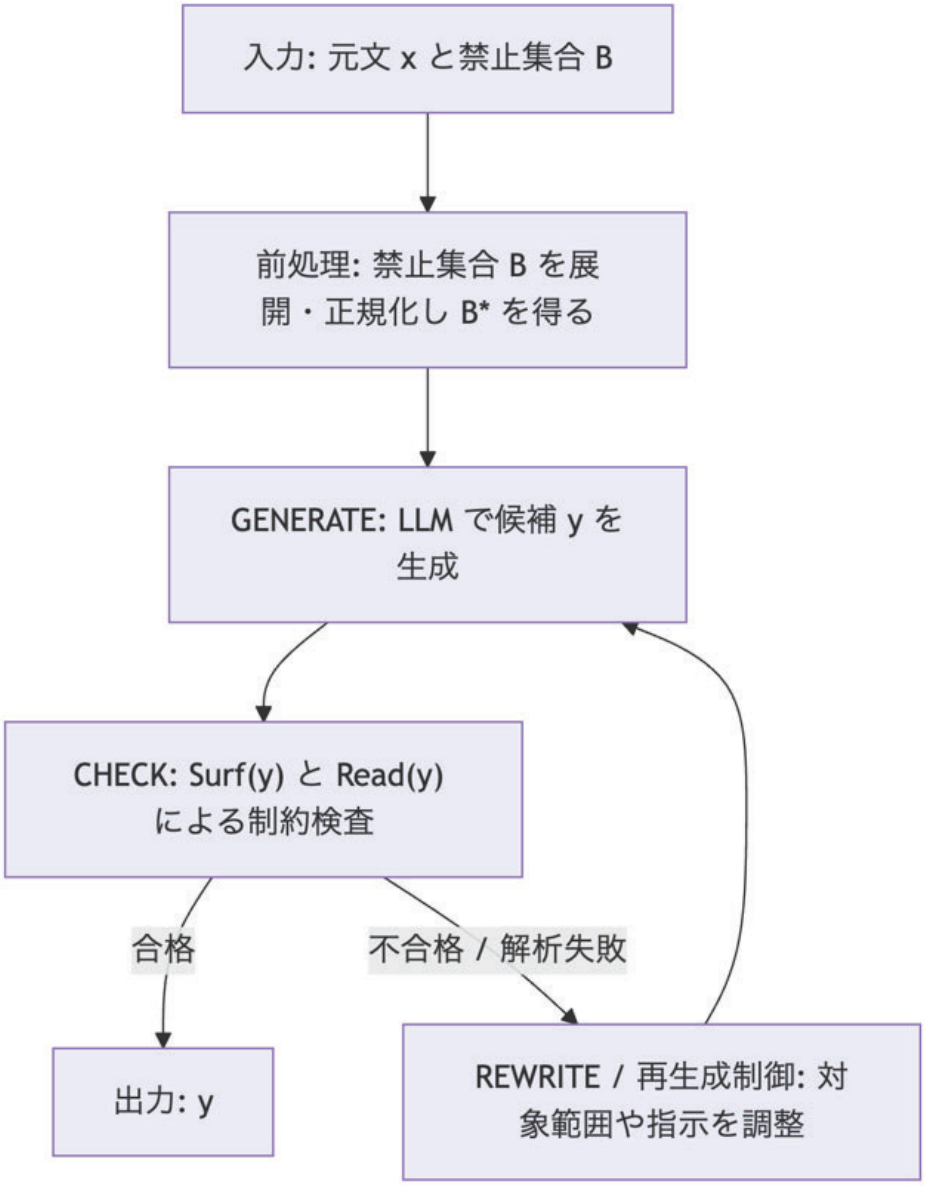

図 1 日本語リポグラム生成パイプライン

4.1.3 逐次生成と一発生成の違い

oneshot と sequential の主な違いは、介入粒度と再試行の構造にある。oneshot は全文を一度に言い換えるため、実装が単純で API 呼び出し回数も少ない一方、違反が残った場合は全文を再生成する以外に介入しづらい。sequential は違反箇所を同定して局所的に書き換えるため、必要な箇所に集中して修正でき、制約遵守率の向上が期待できる。一方で、複数回のチェックと再試行が生じやすく、API 呼び出し回数や実行時間は増加する。

表 1 に、両方式の設計上の差分をまとめる。

表 1oneshot と sequential の比較

観点	oneshot（全文一括）	sequential（局所介入）
介入単位	文全体	トークン／局所区間（必要に応じて拡大）
違反検出	生成後に全文検査	違反箇所の同定を行い、書換え対象にする
再試行の基本形	全文を再生成	局所書き換え→再検査を反復
計算コスト	API 呼び出し回数は少なめ	チェックと再生成が増えやすい
期待される性質	簡潔だが違反が残りやすい	必要箇所へ集中して修正しやすい

4.2 制約チェックアルゴリズム

本節では、第 3 章で定義した表記ベース制約／読みベース制約に基づき、候補文yが禁止集合B^*を含むかどうかを判定する手順を述べる。本研究では、LLM の出力をそのまま信頼するのではなく、外部の決定的な検査（deterministic check）により「違反ゼロ」を保証し、違反が検出された場合は再生成へ回すという設計を採る。

4.2.1 文字レベルチェック

表記ベース制約の検査では、候補文yを正規化した表層文字列$\mathrm{Surf}(y)$を作り、$\mathrm{Surf}(y)$に禁止集合B^*の各要素が含まれるかを調べる。ここでの正規化は、第 3 章の定義に従い、少なくともカタカナをひらがなへ変換する操作を含む。検査は単純な部分文字列探索（または同等の正規表現）として実装できる。

表記ベース制約は、漢字表記には直接反応しないため、読みベース制約と併用することで「表記を変えても読みとしては禁止音を含む」ケースを検出できるようにする。

4.2.2 読みレベルチェック

読みベース制約の検査では、候補文yを形態素解析し、各トークンの読みを取得する。取得した読み（多くの場合カタカナ）をひらがなへ正規化し、連結した仮名列を$\mathrm{Read}(y)$とする。次に、$\mathrm{Read}(y)$に禁止集合B^*の各要素が含まれるかを検査する。

第 3 章で述べた通り、読みが取得できないトークンが存在する場合は安全側に倒し、読みベース制約の判定に失敗（違反）とみなす。このとき、oneshot では全文再生成の対象とし、sequential では局所書き換えの対象範囲を拡大する（例：トークン→文）などの再生成制御により対処する。

4.2.3 複合制約チェック（文字＋読み）

本研究では、表記ベース制約と読みベース制約を併用し、いずれか一方でも違反している場合は違反とみなす。すなわち、$\mathrm{Surf}(y)$と$\mathrm{Read}(y)$の双方が禁止集合B^*を含まない場合にのみ、候補文yを制約遵守と判定する。これにより、表記と読みのいずれの観点でも禁止集合

を回避する出力のみを「成功」として扱える。

4.3　生成アルゴリズム

本節では、候補文を生成する際の LLM への指示（プロンプト設計）と、oneshot／sequential の二方式の生成手順を述べる。両方式は「外部の制約チェックに合格するまで再試行する」という点を共有する一方、介入単位（全文か局所か）と再試行の構造が異なる。

4.3.1　LLM 生成

oneshot と sequential は、いずれも LLM に対して「禁止集合B^*を読みベースで一切含まないこと」を明示し、出力形式を「書き換え後の文のみ」に限定する。禁止集合は、展開・正規化後のB^*を提示する（例：行指定を展開したひらがな列）。

また、プロンプトには少なくとも次の要素を含める。さらに、評価を容易にするため、出力は 1 文（または 1 段落）のみとし、説明文や箇条書きなどの付帯情報を混ぜないように指示する。

- 元文x（書き換え対象）
- 禁止集合B^*（ひらがな列）
- 禁止条件が「読み（仮名列）」に対して課されること
- 日本語として自然であり、元文の主要な意味を大きく損なわないこと

さらに、評価を容易にするため、出力は 1 文（または 1 段落）のみとし、説明文や箇条書きなどの付帯情報を混ぜないように指示する。

生成パラメータ（temperature、top_p など）は方式間で揃え、方式差がプロンプトや推論手順に起因するように統制する。具体的な設定値は第 6 章で述べる。

4.3.2　Oneshot 方式

oneshot 方式では、入力文xと禁止集合B^*をまとめて LLM に与え、全文を 1 回で書き換えた候補文yを得る。得られたyに対して制約検査を行い、制約を満たす場合は出力として採用する。制約違反が検出された場合は、再生成制御により再試行を行う。

再試行では、禁止集合の強調、出力形式の再確認、温度の調整など、生成の安定性に影響する条件を変更しうるが、方式間比較のため、実験では再試行の有無や回数を条件として明示する。oneshot は、失敗時に「どこが違反か」を局所的に修正しづらいため、再試行は基本的に全文を対象として同一タスクを繰り返す形になる。本研究の方式比較では、条件の公平性のため、実験では生成パラメータを固定し、同一条件で再試行回数のみを制御する。

4.3.3　Sequential 方式

sequential 方式では、入力文xを形態素解析してトークン列を得た上で、各トークンについ

て表記および読みのどちらかが禁止集合B^*に違反しているかを判定し、違反トークンを同定する。違反トークンが見つかった場合に限り、その箇所に局所的に介入して書き換えを行う。

書き換えでは、対象トークンを含む文（または近傍の文脈）をプロンプトに含め、「この部分を禁止集合に違反しない表現へ言い換える」ように LLM に指示する。生成した置換候補を適用した後、文全体（または対象区間）に対して再度制約検査を行う。違反が残る場合は、所定回数の範囲で再試行し、それでも解消しない場合は対象範囲を拡大する（例：トークン→句→文、あるいは全文の再生成へフォールバック）といった戦略をとる。

この方式の特徴は、「違反箇所が見つかったときだけ」局所的な再生成を行う点にある。これにより、全文を丸ごと再生成するよりも、制約遵守のための介入を必要箇所へ集中できることが期待される。

4.4 再生成アルゴリズム

本節では、制約違反や解析失敗が生じた際に、再生成（再試行）をどのように駆動するかを述べる。本研究の枠組みは「生成→検査」を基本単位とし、検査で失敗した場合のみ再試行を行う。これにより、生成モデルが出力した文を必ず外部で検証し、違反を残したまま採用しない運用を実現する。

4.4.1 失敗ケースの判定

本研究の枠組みでは、次の場合に「失敗」と判定し、再生成（再試行）を行う。

1. 表記ベース制約または読みベース制約に違反した場合
2. 読みが取得できないトークンが存在し、読みベース制約の判定に失敗した場合
3. 出力形式が規約に違反し、評価に使用できない場合（例：説明文を混ぜる）

4.4.2 再生成戦略

再生成戦略は方式により異なる。oneshot では全文再生成を基本とし、制約違反が出た場合は「条件を明示した同一タスクを繰り返し解かせる」形で再試行する。一方、sequential では局所書き換えの反復を基本とし、違反箇所が解消しない場合に限って対象範囲を段階的に拡大する（トークン→句→文→全文）という方針を採る。局所介入での修正が困難な例（助詞や活用語尾が禁則に抵触する例など）では、早期に「文単位の再構成」へ切り替えることが有効であると考えられる。

4.4.3 停止条件

いずれの方式でも、停止条件（最大試行回数など）を満たした場合は、制約を満たす出力が得られなかった例として記録し、評価時には失敗として扱う。sequential では、局所反復

の無限ループを避けるため、下記のような階層的な上限を設ける。

- 局所での最大試行回数
- 文単位での最大試行回数
- 文全体での最大反復回数

第5章　実装

本章では、第 4 章で述べた日本語リポグラム生成の枠組みを、実際のシステムとしてどのように実装したかを概説する。本研究の成果物は次の三つとして整理できる。

1. 日本語リポグラム生成ロジック
2. バッチ評価と可視化のパイプライン
3. ブラウザから試せる Web アプリ

詳細なアルゴリズムや評価指標の定義は、第 3〜4 章および第 6 章で述べる。

5.1　実装の設計方針

実装の基本方針は、「生成ロジック」と「利用形態（実験／デモ）」を分離し、同一の生成ロジックをバッチ評価と Web アプリの双方から呼び出せるようにすることである。これにより、実験で得られた失敗例を Web アプリ上で再現して観察するなど、分析と試行を往復しやすくなる。

また、本研究は読みベース制約を主制約として扱うため、形態素解析によりトークンの読みを取得し、ひらがなへ正規化した上で禁止文字の有無を判定する処理を、生成と評価の両方で共通に用いる。

5.2　使用技術（ライブラリ）

日本語リポグラム生成の実装では、LLM の呼び出しに OpenAI API クライアントを用い、日本語の読み取得には UniDic を用いた形態素解析（fugashi）を利用する。実験の集計・統計処理には pandas と scipy、可視化には matplotlib／seaborn を用いる。Web アプリは Streamlit により実装した。API キー等の設定は環境変数ファイルを通じて管理する。

5.3　バッチ評価パイプラインの概要

第 6〜7 章の実験では、入力データ（元文と禁止集合の組）を CSV として管理し、これに対して生成方式を切り替えながら一括で書き換えを行い、指標を計算して結果を保存する。保存した結果をもとに成功率や補助指標を集計し、グラフを生成して結果報告へ用いる。実験条件（データセット、モデル設定、生成パラメータ、統計的検定）は第 6 章で詳述する。

5.4　Web アプリケーションの概要

Web アプリは、試行とデモを目的とし、ブラウザ上で元文と禁止文字を入力して生成結

果を確認できるようにした。画面上では入力と出力を見比べられるように表示し、必要に応じて詳細表示（逐次処理のログ）も確認できる。設計上は、Web アプリを「UI 層」として薄く保ち、生成そのものは共通の生成ロジックを呼び出すことで、バッチ評価と同一の実装に基づいて挙動を観察できるようにしている。

図 2 に、Web アプリを含むシステム構成（入力→生成→評価→表示）の全体像を示す。

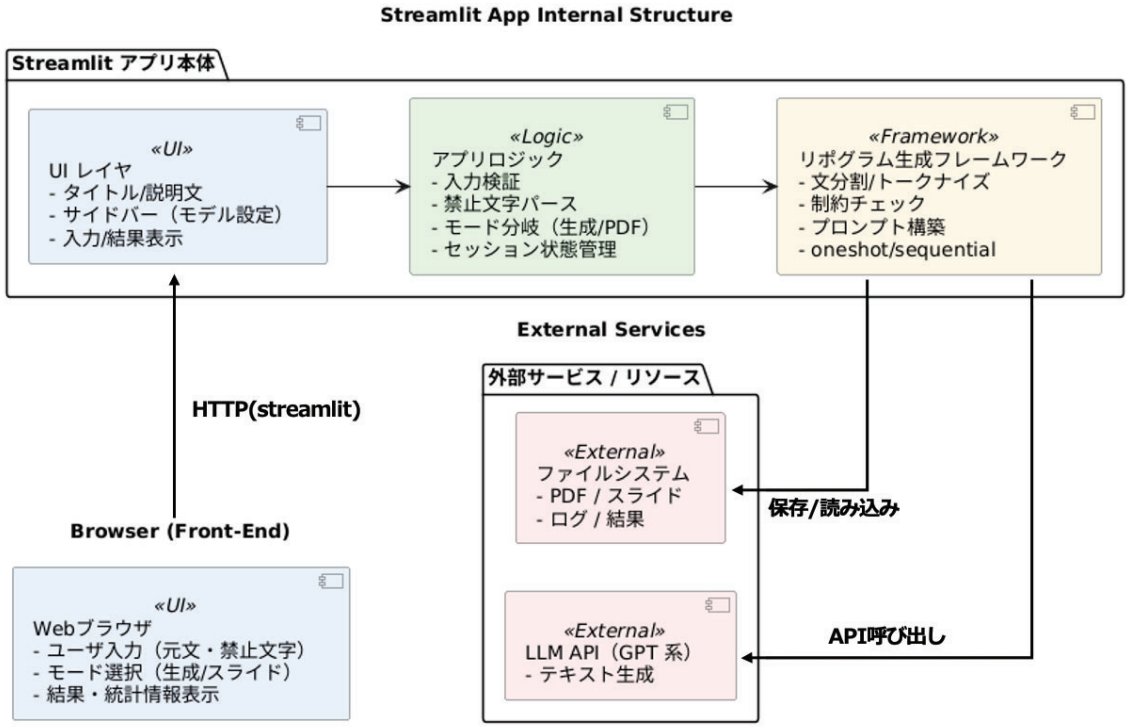

図 2web アプリのシステム構成図

図 2 の上段は Streamlit アプリ内部を、UI レイヤ、アプリロジック、生成フレームワークの 3 層に分けて示している。UI レイヤでは入力（元文・禁止文字）と結果表示を担い、アプリロジックでは入力検証や禁止文字のパース、表示モードの切替、セッション状態管理などを行う。生成フレームワークは、第 4 章で述べた枠組み（文分割・トークナイズ、制約チェック、プロンプト構築）に基づき、LLM を呼び出してリポグラム文を生成する。

図 2 の下段は外部サービス／リソースとの関係を示している。生成フレームワークは外部の LLM API（GPT 系）に問い合わせてテキストを生成し、アプリは必要に応じてファイルシステム上の資料（PDF／スライド）や結果（ログ）を参照して表示する。なお、生成方式（oneshot／sequential）は生成フレームワーク側で切り替え可能な設計とし、実験（第 6〜7 章）では方式差を比較する。

なお、現時点の Web アプリは研究用途（試行・例収集）を想定しており、主観評価実験に転用する場合は、出力の永続保存、参加者の匿名化、A／B 提示の順序制御などの追加機能が必要となる。これらは第 9 章で今後の課題として述べる。

第6章　実験設定

本章では、第 7 章の結果が再現可能になるように、実験に用いたデータセット、モデル設定、評価指標、および集計・検定手順を具体化する。とくに、本研究は「読みベース制約を必須要件とし、違反ゼロを成功とする」点に特徴があるため、読み取得・正規化・判定規約を明記する。

6.1　モデル設定

本研究では、OpenAI の Chat Completions API を通じて LLM を利用する。比較実験では gpt-4.1-nano を基準モデルとして用い、oneshot と sequential の双方で同一モデルを使用することで、方式差が「枠組み（推論手順）」に起因するように統制する。

意味保持を自動評価するために BERT 類似度などの指標を導入する場合は、使用した日本語埋め込みモデル、プーリング方法、類似度計算（コサイン類似度）を明記する。本論文では、まず制約遵守を主目標として比較を行い、意味類似度指標の導入は今後の課題として位置づける。

生成パラメータは方式間で揃える。本実験では温度を temperature=0.5 に固定し、その他サンプリング設定も固定する（例：top_p は既定値）。これにより、乱数性の影響を抑えつつ、oneshot／sequential の差を比較する。

6.2　データセット

本節では、実験に用いる入力データ（元文と禁止集合の組）の作り方と性質を述べる。評価の解釈は入力データの分布や難易度に影響されるため、出典・抽出条件・制約条件（easy／medium）を明確にする。

6.2.1　元文データ

データは Tatoeba 由来の日本語例文コーパスから抽出した。まず、文字数 20〜60 程度で、ひらがなまたは漢字を含む文を条件として 200 文をサンプルし、これをベース文集合とした。その後、各ベース文に対して禁止集合を付与して制約付きパターンを展開し、評価用データセット（465 パターン）を構成した。

表 2 に、ベース文集合と評価用データセットの規模と文長統計を示す。評価用データセットでは、禁止集合に該当する仮名が表記に含まれる場合のみパターンとして採用するため、結果として 195 文が評価対象に含まれる。

表 2 データセット概要

データ	行数	ユニーク文数	文字数(平均／中央値／最小／最大)	備考
ベース文集合	200	200	25.0／23／20／58	Tatoeba から抽出
評価用データセット	465	195	25.4／23／20／58	制約付き展開後

6.2.2　禁止文字設定

禁止集合は、単音禁止（easy）と行禁止（medium）の 2 水準を用意する。easy では 1 文字のみを禁止し、medium では行単位で 5 文字をまとめて禁止する。表 3 に、禁止集合の種類と、評価用データセットにおける出現件数（パターン数）を示す。

表 3 禁止集合の種類と件数

制約強度	禁止集合（banned_chars）	件数
easy	い	115
easy	さ	34
easy	ら	46
medium	あ、い、う、え、お	160
medium	か、き、く、け、こ	110

本実験では、制約強度（easy／medium）を主要な条件として扱う。文長やジャンルなどの層別は、将来の拡張として設計可能であるが、本データセットではジャンルは tatoeba に統一されている。より多様なジャンル（ことわざ、物語文など）を含む設計は今後の課題として第 9 章で述べる。

6.2.3　データの例

表 4 に、評価用データセットに含まれる入力例を示す。各行は「元文x」と「禁止集合B」の組であり、評価ではこの組に対して oneshot と sequential の両方式で書き換えを行う。

表 4 入力例（元文＋禁止集合）

元文（例）	禁止集合（例）	制約強度
彼がこういうことを言ったのだと思われる。	い	easy
母の日にお母さんにカーネーションをあげた。	さ	easy
彼は一生懸命やっているからうまくいくだろう。	ら	easy
彼がこういうことを言ったのだと思われる。	あ、い、う、え、お	medium
彼がこういうことを言ったのだと思われる。	か、き、く、け、こ	medium

6.3　評価方法

本節では、同一入力に対して oneshot と sequential を適用し、成功判定（読みベース制約）と補助指標を計算する手順を述べる。評価は「生成→制約検査→指標計算→結果保存」を基本単位とし、これを全入力に対して繰り返す。

6.3.1　意味保持評価

意味保持は品質要件であり、制約遵守（主指標）を満たした上で評価すべき対象である。本論文では、意味保持そのものを自動指標で直接測るのではなく、まず書き換えの大きさ（VRR）や語彙の性質（TTR・n-gram 反復率）を補助指標として計測し、第 8 章で代表例に基づく定性的な議論を行う。意味類似度指標（BERT 類似度など）の導入は今後の課題とする。

6.3.2　文法性評価

文法性・自然性は、最終的には人手評価が必要である。主観評価を行う場合は、A／B 比較（oneshot／sequential）をブラインドで提示し、可読性・自然さ・意味保持を Likert 尺度で評価する設計が考えられる。現時点で主観評価が未実施の場合は、自動指標のみで断定できる範囲を限定し、第 8 章で代表例を用いて議論する。

6.3.3　制約遵守率評価

制約遵守率は読みベース制約を主指標として計測する。具体的には、生成文を形態素解析して得られる各トークンの読み（カタカナ）をひらがなへ正規化し、連結した$\mathrm{Read}(y)$に禁止集合が含まれるかで判定する。読みが取得できないトークンが存在する場合は安全側に倒し、制約判定に失敗（違反）として扱う。

6.3.4　補助指標（VRR／TTR／反復率／時間）

制約遵守以外の性質を捉えるために、補助指標として VRR（語彙置換率）、TTR（語彙多様性）、n-gram 反復率、実行時間を計測する。これらは評価用の実装で算出し、記号トーク

ンを除外した上で計算する（詳細は付録）。VRR は、トークン数が入力と出力で一致しない場合に備え、最長共通部分列（Longest Common Subsequence; LCS）に基づく近似を用いて「どれだけ置換が起きたか」を頑健に見積もる。

6.3.5　ベースライン

比較対象は oneshot と sequential の 2 方式である。oneshot は「プロンプトで禁止集合を与え、全文を一度に書き換える」素朴なベースラインとして位置づける。sequential は、違反箇所の検出と局所介入を行う枠組みとして位置づけ、両者を同一モデル・同一生成パラメータで比較する。

6.3.6　アブレーション（枠組み要素の寄与）

枠組み要素の寄与を切り分けるためには、読みチェックなし（表記のみ）、局所修正なし（全文のみ）、再生成制御なし（1 回生成のみ）などのアブレーションを設計できる。本論文では、まず oneshot／sequential の比較結果を報告し、アブレーションは今後の拡張として実施可能な形で設計方針を整理する。

6.3.7　統計的検定

成功率は同一入力に対する方式比較であるため、対応のある検定により差の有意性を確認する。本研究では、成功フラグ（0／1）について対応のある t 検定を行い、方式差の有無を検討する。連続値の指標（VRR／TTR 等）については、「両方式が成功したペア」に限定した上で、対応のある t 検定により方式差を検討する。多重比較を行う場合は Holm 法などの補正を用いる。

表 5 に、本実験で用いる主要な比較条件と評価指標の対応関係をまとめる

表 5 実験条件と評価指標

項目	設定	位置づけ
データ	評価用データセット（465 パターン）	評価対象
方式	oneshot／sequential	比較条件（主）
主制約	読みベース制約Read(y)	必須要件
主指標	制約遵守率（読み）	主指標
補助指標	VRR、TTR、n-gram 反復率、実行時間	品質・性質の補助
モデル	gpt-4.1-nano（Chat Completions）	統制条件
生成設定	temperature=0.5（固定）	統制条件

6.4　再現性情報

本節では、本実験を再実行するために必要な情報を整理する。LLM を用いるため完全な決定性は得られないが、入力データと条件を固定し、結果（生成文と指標）を CSV として保存することで、第 7 章の集計と図表生成は再現できる。

本研究の実験はバッチ実行により再現できる。評価用データセット（CSV）を入力し、方式とモデル名を指定して生成・評価を行い、結果を CSV として保存する。

図 3 に、バッチ評価の処理フロー（入力→方式別生成→指標計算→結果保存→集計・可視化）を示す。生成はネットワーク（LLM API）に依存するが、指標計算と集計・図表生成はローカルで完結する。

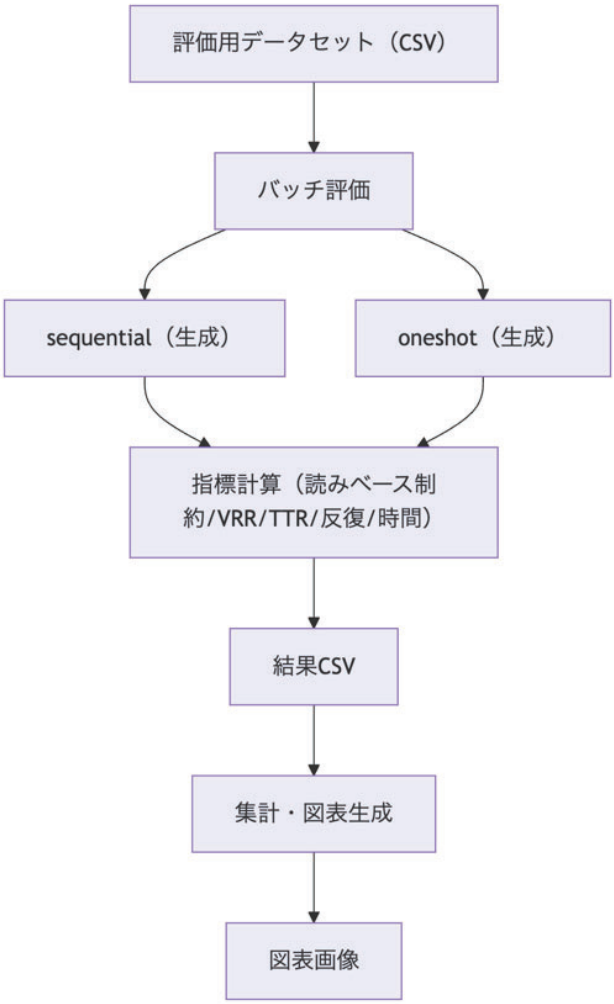

図 3 バッチ評価の処理フロー

LLM 生成は確率的であるため、温度を固定しても完全な再現は保証されない。再現性を高めるためには、同一条件で複数回実行して平均・信頼区間を報告する方法があるが、本論文ではまず方式差の大きさを明確にすることを優先し、単回実行の結果を中心に報告する。

第7章　実験結果

本章では、第 6 章で定めた実験設定に基づき、oneshot と sequential の比較結果を示す。主指標である読みベース制約遵守率を中心に、補助指標（VRR／TTR、反復率、実行時間）や代表例を用いて、各方式の得意・不得意を整理する。

7.1　制約遵守率（主指標）

本研究の主指標は読みベース制約遵守率であり、「生成文の読みRead(y)に禁止集合が一切含まれない」出力の割合として定義する。本節では、方式別（oneshot／sequential）および制約強度別（easy／medium）に成功率を示す。

まず、全 465 文（各方式 465 件）に対する成功率を表 6 生成方式別の読みベース制約遵守率に示す。成功率は、保存した実験結果ログ（CSV）の制約違反フラグに基づき集計した。

表 6 生成方式別の読みベース制約遵守率

方式	成功率	成功数／総数
oneshot	12.5%	58／465
sequential	91.6%	426／465

図 4 は、表 6 と同じ集計を棒グラフとして示したものである。

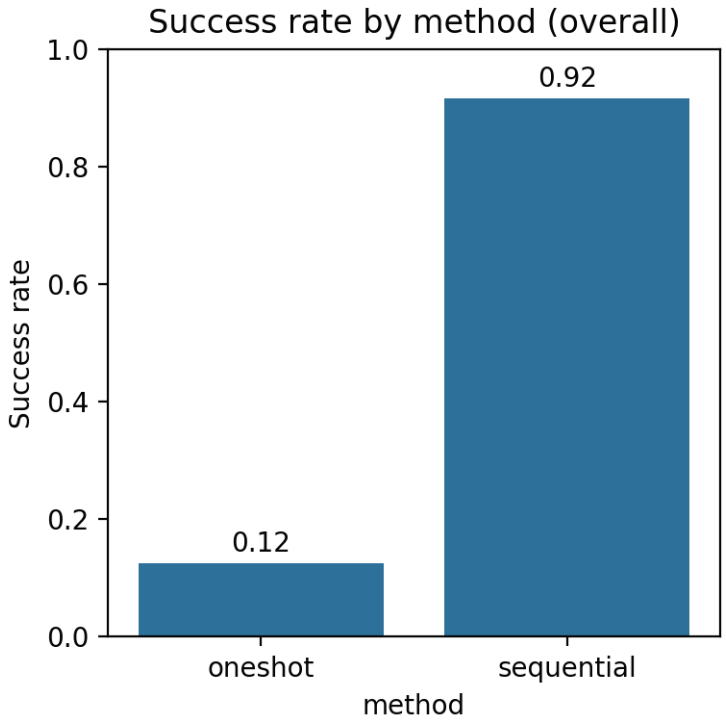

図 4 生成方式別の読みベース制約遵守率

次に、制約強度（easy：単音禁止、medium：行禁止）ごとの成功率を表 7 および図 5 に示す。とくに medium 条件では oneshot の成功率が大きく低下し、厳しい制約下で sequential の優位性が顕著であることが分かる。

表 7 制約強度×生成方式別の読みベース制約遵守率

制約強度	oneshot（成功率／件数）	sequential（成功率／件数）
easy	27.7%（54／195）	99.0%（193／195）
medium	1.5% （4／270）	86.3%（233／270）

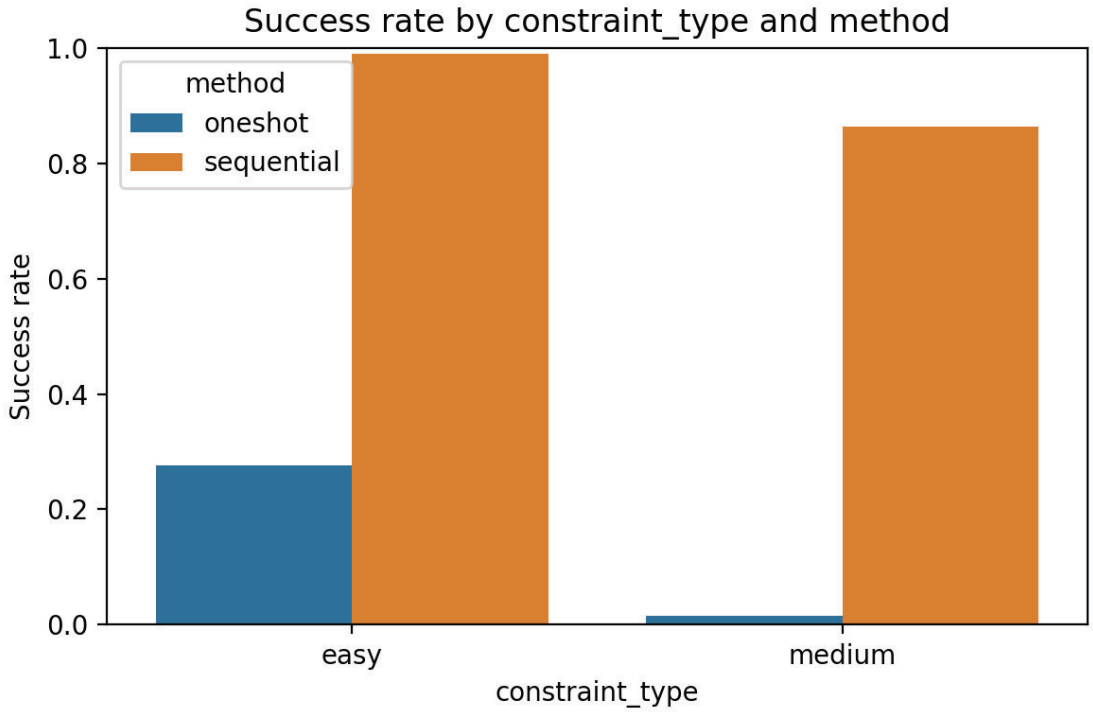

図 5 制約強度×生成方式別の読みベース制約遵守率

次に、「同じ入力に対して両方式を試したとき、どちらが成功しやすいか」を見るために勝ち負けを集計する。本実験では、各入力（元文xと禁止集合Bの組）に対して、oneshot と sequential をそれぞれ 1 回ずつ適用しているため、同じ入力に対する 2 つの結果を 1 組として比較できる。表 8 は、この 1 組ごとに成功フラグ（0／1）を比較し、「sequential のみ成功」「oneshot のみ成功」「同じ結果（両方成功または両方失敗）」に分類した結果である。oneshot のみが成功した例は観測されず、多くの入力で sequential が優位であることが分かる。

表 8 同一パターンにおける勝ち負け

区分	件数	割合
sequential のみ成功	368	79.1%
oneshot のみ成功	0	0.0%
同じ結果（両方成功 or 両方失敗）	97	20.9%

成功フラグ（0／1）に対する対応のある t 検定では、方式差は統計的に極めて有意であった（$t(464) = 41.96$、$p \approx 5.0 \times 10^{-160}$）。

7.2　意味保持（補助指標と代表例）

制約遵守を満たしたとしても、元文の主要な意味が損なわれていれば実用性は低い。本研究では意味保持を品質要件として位置づけるが、本論文では直接的な意味評価（意味類似度指標や人手評価）は未実施である。そこで、本章では VRR などの補助指標と代表例（7.7 節）を中心に、意味保持の観点から定性的に議論し、直接的な評価指標の導入は第 9 章の今後課題で述べる。

7.3　文法性・自然性

本節では、出力が日本語としてどの程度自然に読めるか（文法性・自然性）を整理する。文法性・自然性は最終的に人手評価が必要であるが、本論文では人手評価を未実施のため、代表例（7.7 節）と補助指標（7.4 節の反復率など）を手がかりに、定性的に述べる。詳細な考察は第 8 章で行う。

7.4　書き換えの性質（VRR／TTR／反復）

本節では、制約遵守以外の性質として、「どれだけ大きく言い換えたか」や「語彙の多様性」を補助指標で捉える。ここでは語彙置換率（VRR）と type-token ratio（TTR）を中心に、成功ケースに限定した比較や、VRR と成功・失敗の関係を示す。

図 6 は、両方式がともに制約を満たした入力（58 ペア）に限定し、方式ごとの VRR の分布を示す。分布の中心が oneshot の方が高く、oneshot がより大きな言い換えを行う傾向が見て取れる。

図 7 は、両方式が成功したケースにおいて、VRR と TTR の関係を方式別に可視化したものである。TTR は両方式で大きく変わらない一方、VRR は oneshot 側に高い点が多く、図 6 の傾向と整合的である。

図 8 は、VRR と制約遵守成否（成功／失敗）の関係を示す。VRR が高い（大きく言い換える）ほど制約を満たすのが難しくなる傾向があり、成功率（7.1 節）と書き換え量の間にトレードオフがあることを示唆する。

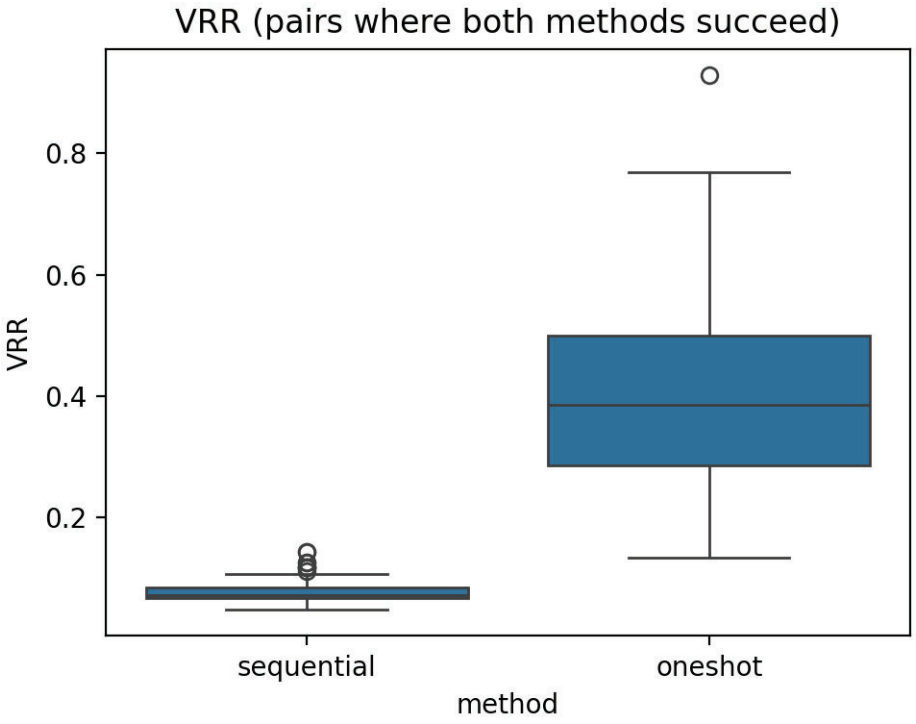

図 6 両方式が成功したケースにおける VRR の分布

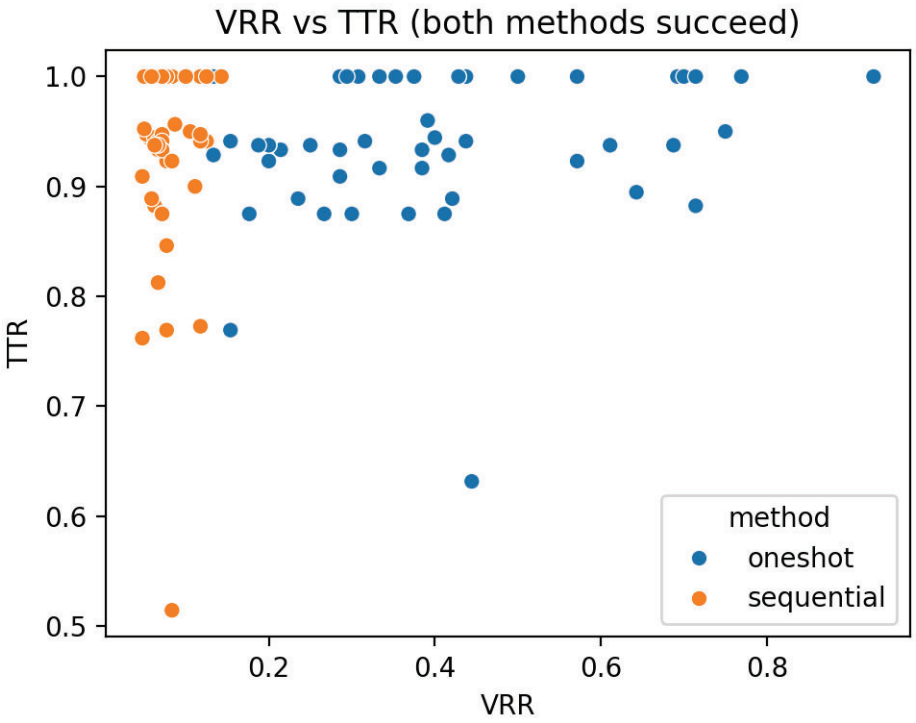

図 7 両方式が成功したケースにおける VRR と TTR の散布図

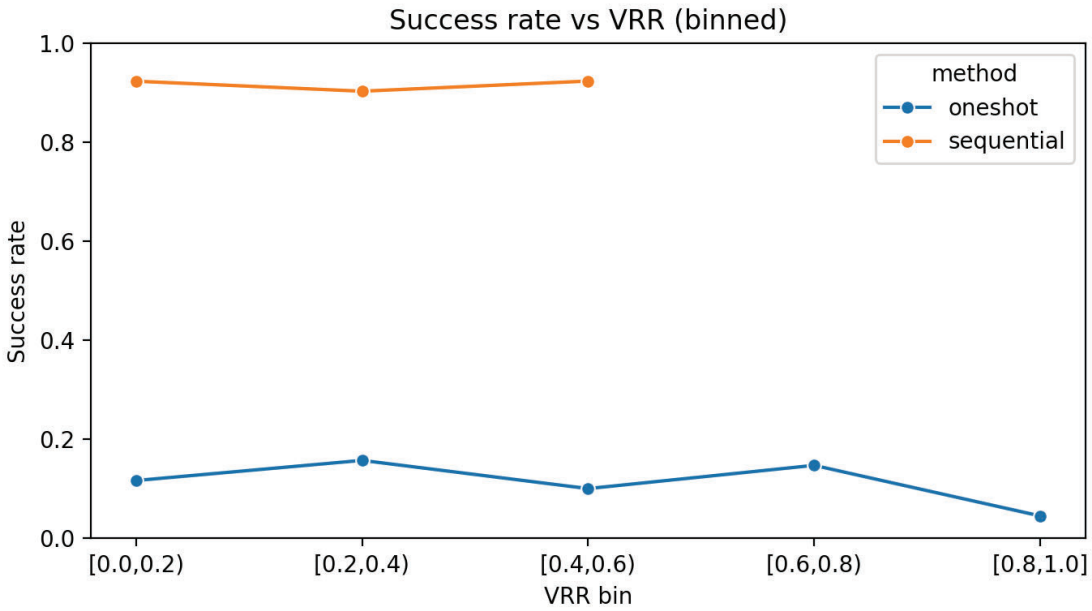

図 8VRR と制約遵守成否の関係

両方式が成功した 58 ペアに限定し、VRR と TTR を対応のある検定で比較すると、VRR は oneshot の方が有意に高く（より大きく言い換える傾向）、TTR の差は有意とはいえない結果となった（表 9）。

表 9 両方式成功ペアにおける指標差の検定

指標	比較対象	結果
VRR	sequential vs oneshot	$t(57) = -13.74$、$p \approx 9.8 \times 10^{-20}$
TTR	sequential vs oneshot	$t(57) = -1.40$、$p \approx 0.166$

補助指標のうち反復率（n-gram repetition rate）は、生成文における不自然な繰り返しの有無を確認するために用いる。本実験では、多くの出力で反復率は 0 であり、bi-gram 反復が観測された割合は sequential が 9.9%、oneshot が 5.6%であった（表 10）。tri-gram 反復が観測された割合はいずれも 4%未満であり、強い反復が頻出する状況ではなかった。

表 10n-gram 反復率の要約

方式	bi-gram 反復率（平均）	bi-gram 反復ありの割合	tri-gram 反復率（平均）	tri-gram 反復ありの割合
oneshot	0.006	5.6%	0.003	2.4%
sequential	0.010	9.9%	0.004	3.7%

7.5　計算コスト

本節では、方式間の計算コストを比較する。逐次生成方式は局所介入と再検査を反復するため、制約遵守率が高い一方で、実行時間が増えやすい。本論文では、1 文あたりの実行時間（秒）を用いて、成功率とのトレードオフとして整理する。

表 11 に、方式ごとの 1 文あたり実行時間の要約（平均・中央値）を示す。逐次生成は再試行や局所介入を伴うため、成功率改善の代償として計算コストが増加する傾向がある。

表 11 方式別の実行時間（1 文あたり）

方式	平均（秒）	中央値（秒）
oneshot	0.696	0.615
sequential	7.188	2.631

7.6　ベースライン・アブレーション

本研究の貢献は「枠組み」の提案にあるため、枠組み要素の寄与を切り分けて示すことは重要である。一方で、本論文では oneshot／sequential の比較を優先し、読みベース制約チェックのみ（表記ベース）や局所修正なし（全文再生成のみ）といったアブレーションは未実施である。アブレーション設計の方針は第 6 章で述べ、今後の課題として第 9 章で扱う。

7.7　失敗分析（定量＋定性）

本節では、失敗例を分類し、「どの条件で何がボトルネックになるか」を明らかにする。とくに、日本語では助詞や活用語尾が制約に衝突しやすいこと、行禁止のように禁止集合が大きい場合に迂回表現の探索が難しくなること、未知語・解析の揺れにより読み判定が不安定になることが、失敗要因になりうる。代表例は実験ログから抽出し、元文／禁止集合／各方式の出力を並置して示す。

図 9 は、行禁止（あ行）の例である。oneshot の出力では、「こう」「ないよう」「推測」など日常的な表現が含まれ、読みベースでは禁止集合（あ行）に該当する仮名が混入して制約違反となっている。一方、sequential は制約を満たす出力を得ているが、局所的な言い換えを積み重ねた結果として語の重複（例：「述べ…述べ…」）や不自然な言い回しが残り、制約遵守と自然さのトレードオフが表れている。

図 10 は、行禁止（か行）の例である。oneshot の出力は、「推測（すいそく）」のように読みが禁止集合（か行）に触れやすい語を含み、制約違反となっている。sequential は制約を満たすように表現を回避できているが、その過程で「彼」→「あの人」のような語彙変更や、「例えば」の挿入などが生じ、語順・語彙が大きく変化している。厳しい行禁止では、助詞や語彙の選択肢が狭まり、大きな言い換えが必要になる場合があることが分かる。

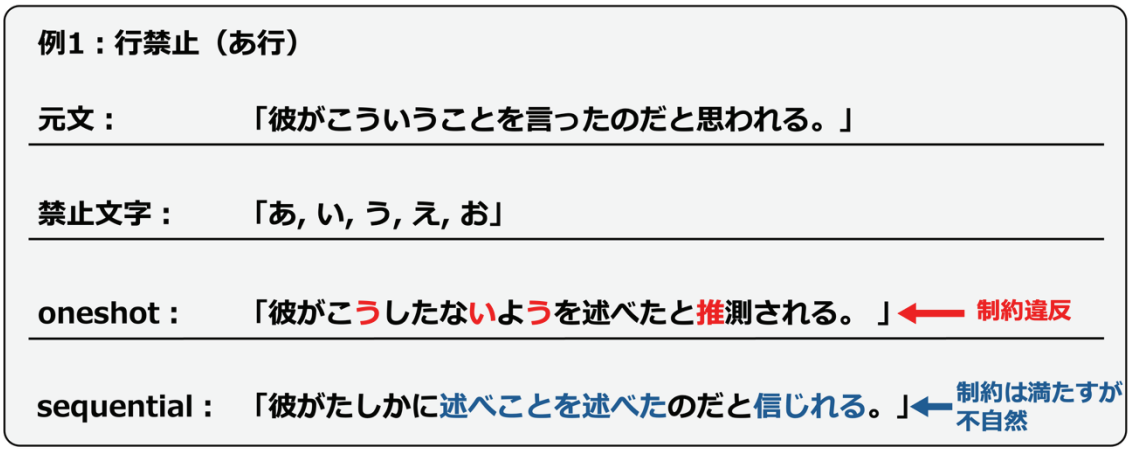

図 9 代表例 1（元文／制約／出力の比較）

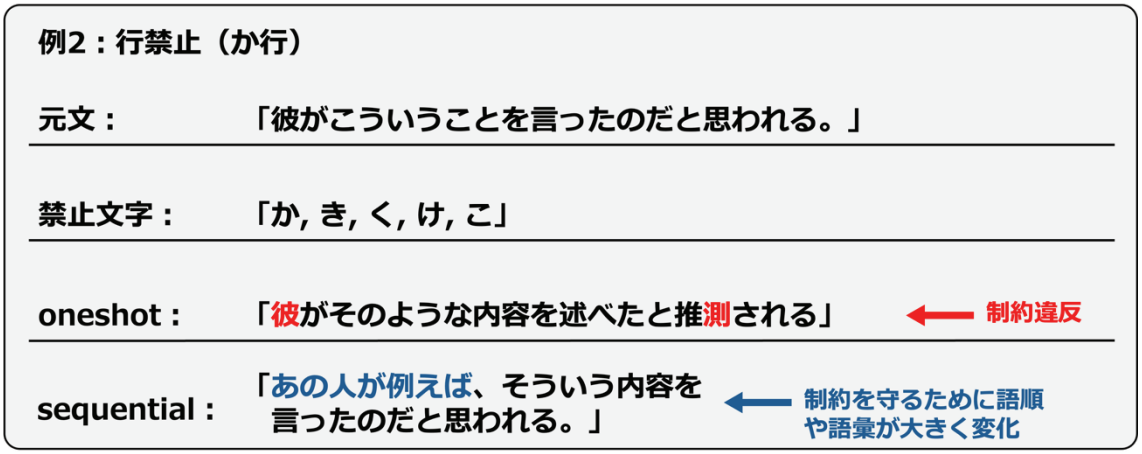

図 10 代表例 2（元文／制約／出力の比較）

第8章　考察

本章では、第 7 章で得られた結果が「なぜそうなったか」を、本研究が提案する枠組み（読みベース制約の検査と、検査に基づく再生成・局所修正）および日本語の言語特性の観点から解釈する。あわせて、実験で明らかになった限界と、今後の設計指針を整理する。なお、本論文では人手評価は未実施であるため、考察は主指標（制約遵守率）と補助指標（VRR／TTR、反復率、実行時間）、および代表例の観察に基づいて行う。

8.1　制約遵守率の考察

第 7 章の結果（表 6、表 7、表 8）から、読みベース制約遵守率は sequential が oneshot を大きく上回った。とくに行禁止（medium）で差が顕著であり、oneshot はほとんど成功しない一方、sequential は高い成功率を維持している。これは、リポグラムが「1 文字でも混入すれば失敗」という厳格な負制約であり、生成モデルに対するプロンプト指示だけでは違反を完全に抑え込みにくいことを示す。

oneshot が破りやすい主因は、生成過程で禁止音（読み）を逐次的に監視・修正していない点にある。表層では問題がなく見えても、読みで禁止音が混入することがあり、とくに日本語では漢字表記が読みの混入を隠してしまう。行禁止のように禁止集合が大きい場合は回避すべき仮名が増えるため、禁則を満たしながら自然な文を一度で生成する難度が急上昇し、微小な違反が残りやすい。

一方で sequential は、「生成→検査→局所修正→再検査」という反復を前提とし、失敗が起こる箇所を逐次的に潰す設計である。読みベースの検査を毎ステップで行い、違反を含むトークン（または局所区間）に介入することで、最終的に違反ゼロへ到達する確率が高くなる。第 7 章の勝ち負け集計（表 8）で「sequential のみ成功」が多数を占め、oneshot のみ成功が観測されなかったことは、検査と局所介入が制約遵守に対して決定的に効いていることを裏づける。

8.2　日本語に特有の影響

本研究のタスクでは、制約が「ひらがな（音）」で与えられるため、日本語の表記体系（漢字・かな混在）と形態素構造が直接的に難しさへ影響する。まず、同じ意味内容でも語彙選択の自由度は制約により大きく制限される。行禁止では基本語彙や助詞・活用語尾に含まれる頻出音がまとめて禁止されるため、単純な置換では回避できず、文全体の再構成が必要になる。

また、読みベース制約では形態素解析器の出力（読み）に依存するため、未知語・固有表現・解析揺れが判定結果に影響しうる。読みが取得できない場合は安全側に倒して違反とみ

なす規約としているため、解析が不安定な入力では成功率が下がる可能性がある（本実験では内訳の定量分析は行っていない）。この点は、読みベース制約の厳密性と引き換えに生じる実装上の制約である。

8.3　意味保持の難しさ

意味保持は品質要件であるが、本研究では直接的な意味類似度指標や人手評価を実施していないため、第 7 章の補助指標と代表例に基づき、間接的に議論する。まず、VRR は書き換え量の近似であり、値が高いほど表現の変更が大きいことを示す。第 7 章では、両方式が成功したケースに限定すると oneshot の VRR が有意に高く（表 9、図 6）、oneshot がより大きな言い換えを行う傾向が確認された。これは、成功した場合の oneshot が大胆な言い換えで禁則を回避できる一方、その分だけ意味逸脱のリスクも高くなる可能性を示唆する。

一方で、VRR と成功／失敗の関係（図 8）は、「大きく言い換えるほど制約を満たすのが難しい」傾向を示しており、制約遵守と表現自由度の間にはトレードオフがある。sequential は局所修正を積み重ねるため、結果として VRR が低めになりやすく、元文に近い表現を保ったまま禁則を満たす方向に働く。ただし、局所修正の積み重ねは冗長さや表現の重複を生みうるため、意味保持・自然さの評価は代表例と合わせて慎重に行う必要がある。

8.4　手法の限界

第一に、計算コストの増大である。第 7 章の実行時間（表 11）から、sequential は oneshot より大幅に遅い。これは、検査と局所介入を反復する設計の必然であり、成功率向上の代償として現れる。実運用では、許容遅延や API 利用コストに応じて、反復回数の上限や介入粒度を調整する必要がある。

第二に、長文・段落への拡張性である。本研究は文単位の生成を対象としており、入力が長くなるほど、違反箇所の探索や局所修正の影響範囲が拡大し、反復回数が増える可能性がある。長文では、局所修正が文全体の整合性（照応や時制など）に波及しやすく、単純な逐次介入では自然さが損なわれる恐れがある。

第三に、読みベース制約の解析器依存である。読み取得に失敗した場合は違反とみなすため、解析の揺れや未知語が多い入力では成功率が過小評価される可能性がある。解析器の変更や辞書拡張、読み推定の補助などは、枠組みの頑健性を高める上で重要な改良点である。

8.5　設計指針（枠組み提案としてのまとめ）

第 7 章の結果を踏まえると、日本語リポグラム生成においては、少なくとも次の設計指針が得られる。

- 厳格な負制約（違反ゼロ）を満たすには、生成後の検査だけでなく、検査に基づく再

生成・修正を組み込む必要がある。

- 禁止集合が大きい条件（行禁止）では、oneshot だけでは成功率が急落しやすく、sequential のような局所介入型の枠組みが有効である。
- 一方で sequential は計算コストが増えやすいため、許容遅延に応じて反復回数や介入単位（トークン／句／文）の設計を調整する必要がある。
- 意味保持・自然さの評価は、制約遵守を満たした出力に限定して行うのが望ましい。書き換え量（VRR）や反復率は、品質の兆候を与える補助指標として有用であるが、より直接的な意味評価指標の導入は今後の課題である。

第9章　奨学金の活用

本章では、本研究を進めるにあたり iTL 先端的プロジェクト奨学金をどのように活用したかについて述べる。本奨学金を活用することで、研究成果の学会発表、リポグラム生成システムの構築、および文章生成実験のための API の利用を行うことができた。以下では、これらの取り組みについて具体的に説明する。

9.1　学会発表

本研究は第 61 回サイバーワールド研究会にて学会発表を行なった。発表では、日本語リポグラム自動生成という課題設定と、本研究で提案する生成枠組みについて報告した。学会発表を通じて研究内容を外部に発信するとともに、さまざまなフィードバックや議論を通じて本研究の課題や今後の展望について理解を深めることができた。

9.2　システム構築

本研究では、日本語リポグラム生成の枠組みを検証するため、文章生成システムの構築を行った。本奨学金は、研究に必要な計算環境の整備や開発環境の構築に活用した。

また、本奨学金はシステム開発に必要な知識を習得するための専門書の購入にも活用した。これにより、文章生成や自然言語処理に関する理解を深めながら、研究に必要なプログラムの設計および実装を進めることができた。

9.3　API 利用

文章生成の実験においては、LLM モデルの外部 API を利用した。本奨学金を活用することで、API 利用に伴う費用を賄い、大規模言語モデルを用いた文章生成実験を行うことができた。

図 11 学会発表の様子

第10章 結論と今後の課題

本章では、本研究で提案した日本語リポグラム生成の枠組みを総括し、実験から得られた知見を整理したうえで、今後の課題と発展可能性を述べる。

10.1 本研究のまとめ

本研究の目的は、「与えられた日本語文xと禁止集合Bに対して、禁止集合を読みレベルで一切含まない日本語文yを生成する」という日本語リポグラム生成タスクに対し、実用的に制約遵守（違反ゼロ）を達成するための枠組みを提案することである。日本語では漢字表記により禁止音の混入が表層から見えにくいことがあるため、本研究は読みベース制約（Read(y)）を主制約として採用した。

提案枠組みでは、次の要素を中核として位置づけた。

1. 読みベースの制約検査検
2. 査に基づく再生成・局所修正
3. 再検査の反復

枠組みに基づく具体的な方式として、oneshot（全文を一度に生成）と sequential（違反検出と局所介入を反復）を同一モデル・同一条件で比較し、読みベース制約遵守率を主指標として評価した。

10.2 得られた知見（RQ1〜RQ3）

第 7 章の結果と第 8 章の考察を踏まえ、本研究の知見を研究課題（RQ）に沿って整理する。

RQ1（制約遵守率）

読みベース制約遵守率は sequential が oneshot を大きく上回った。とくに行禁止（medium）のような厳しい条件では oneshot の成功率が急落し、検査と局所介入を組み込んだ枠組みの有効性が確認された。

RQ2（書き換えの性質）

両方式が成功したケースに限定すると、oneshot は sequential より VRR が有意に高く、より大きな言い換えを行う傾向が見られた。一方で、TTR の差は有意ではなく、語彙多様性そのものは大きく変わらない結果であった。

RQ3（条件依存）

禁止集合が大きい（行禁止）ほど制約遵守が難しくなり、方式差が拡大した。また、書き換え量（VRR）が大きいほど成功が難しくなる傾向が観察され、制約遵守と表現自由度の間にトレードオフがあることが示唆された。

加えて、計算コストの観点では sequential の実行時間が oneshot を大きく上回り、制約遵守率の改善と計算コストの間に明確なトレードオフがあることが確認された。

10.3 今後の課題

本研究は制約遵守（違反ゼロ）の達成を主目標として枠組みを検討したが、品質面・評価面・運用面には改善余地がある。主要な課題を以下にまとめる。

10.3.1 意味保持・自然さの評価と向上

本論文では人手評価や意味類似度指標を導入していないため、意味保持・自然さについては代表例に基づく定性的な議論に留まった。今後は、人手評価の設計（評価項目、提示方法、評価者間一致の確認）を整備したうえで、制約遵守を満たした出力に限定して品質評価を行う必要がある。

あわせて、意味保持を補助する自動指標（意味類似度など）も有用である。ただし、類似度が高いことが必ずしも意味保持を保証しない（否定、数量、固有表現の差など）点に注意し、代表例と併用しながら解釈することが望ましい。

10.3.2 ハイブリッド方式（候補生成＋検査＋選別）

oneshot は成功時に大きな言い換えを行える一方、厳しい制約下では成功率が低い。sequential は成功率が高い一方、計算コストが増えやすい。今後の方向性として、複数候補を生成して制約検査でふるい分け、必要な場合のみ局所修正へ移行するハイブリッド方式が考えられる。これにより、成功率と計算コストのバランスを改善できる可能性がある。

10.3.3 読みベース制約の頑健化

読みベース制約は形態素解析器の出力に依存するため、未知語・固有表現・解析揺れが判定に影響しうる。辞書拡張や解析器の切り替え、読み推定の補助などにより、読み取得の失敗を減らすことが課題である。また、禁止集合の入力形式（行指定だけでなく段・濁音半濁音・拗音など）を拡張し、ユーザが意図する「禁止音」を柔軟に指定できるようにすることも有用である。

10.3.4 長文・多様ドメインへの拡張

本研究は文単位の生成を対象とし、例文コーパスを中心に評価した。今後は、ことわざ・物語文・対話文など多様なジャンルを含むデータ設計や、段落レベルの生成への拡張が必要

である。長文では局所修正が文全体の整合性に影響しやすいため、介入単位（トークン／句／文）の設計や、再生成戦略の見直しが求められる。

10.3.5 計測・再現性の強化

本研究では実行時間を主要なコスト指標として扱ったが、将来的には API コール回数や再試行回数などの運用上のコストも記録し、方式間の比較をより明確にすることが望ましい。さらに、同一条件で複数回実行し、平均や信頼区間を報告することで、確率的生成に対する再現性を高めることができる。

10.4 発展可能性

日本語リポグラムは制約付き創作としての側面を持ち、生成支援は創作活動に新たな発想を与える可能性がある。たとえば、禁止音を指定して「言い換えの幅」を探索することで、表現のバリエーションを増やす創作支援に応用できる。

また、対話システムや文章生成システムにおいて、不適切表現を含まない出力や特定スタイルに沿った出力を求める場面では、検査と再生成を組み合わせた枠組みは一般の制約付き生成へも展開可能である。さらに、語彙や仮名の制約を利用した言語学習・作文教育（語彙選択の練習、言い換え練習）への応用も考えられる。

加えて、本研究の枠組みは「言い換え」を前提としているため、発話支援にも応用できる。たとえば、音韻上の言いづらさを避ける（特定の音を含む語を避ける）、あるいは言い淀みや繰り返しを抑えるといった目的に対して、発話内容の候補を複数提示し、制約検査でふるい分けたうえで、必要に応じて局所的に修正する支援が考えられる。リアルタイム性が求められる場面では、候補生成の本数や反復回数を制限し、計算コストと品質のバランスを取る設計が重要となる。

謝辞

本研究の遂行にあたり、ご指導・ご助言を賜りました飯尾淳先生に、心より感謝申し上げます。研究の方向性の検討から論文執筆に至るまで、多くの示唆をいただき、本研究をまとめ上げる上で大きな支えとなりました。

また、研究室の皆様には、日頃の議論や発表練習の機会を通じて多くの助言をいただきました。特に、中間発表や研究会でのコメントは、本研究の課題設定と評価方法を見直すきっかけとなり、内容の改善につながりました。ここに感謝の意を表します。

最後に、本研究の遂行にあたり研究活動をご支援いただいた iTL 先端的プロジェクト奨学金ならびに中央大学国際情報学部の関係者の皆様に深く感謝申し上げます。

参考文献

1. G. Perec、 La Disparition、 Éditions Denoël、 1969 年.
2. S. Gehman、 S. Gururangan、 M. Sap、 Y. Choi、 N. A. Smith、 RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models、 Findings of the Association for Computational Linguistics: EMNLP 2020、 pp. 3356–3369、 2020 年.
3. T. Kajiwara 、 Negative Lexically Constrained Decoding for Paraphrase Generation 、 Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)、 pp. 6047–6052、 2019 年.
4. J. Devlin、 M.-W. Chang、 K. Lee、 K. Toutanova、 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding、 Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)、 巻 1、 pp. 4171–4186、 2019 年.
5. T. Shen、 T. Lei、 R. Barzilay、 T. Jaakkola、 Style Transfer from Non-Parallel Text by Cross-Alignment、 Advances in Neural Information Processing Systems、 巻 30、 2017 年.
6. J. Li、 R. Jia、 H. He、 P. Liang、 Delete、 Retrieve、 Generate: A Simple Approach to Sentiment and Style Transfer、 Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)、 巻 1、 pp. 1865–1874、 2018 年.
7. Loujain、 lipogram_e: French lipogram text generation with GPT models、 GitHub repository、 URL: https:／／github.com／Loujainl／lipogram_e(最終参照日 2026／1／9)、 2024 年.