制約・曖昧性・モデルサイズに関する考察

— SLM と LLM および Transformer の性質を中心に —

Y. Matsuda

26 March 2026

1 序論

近年の大規模言語モデル（LLM）は、その高い表現力により多様なタスクを統一的に扱うことを可能にした。一方で、計算資源の大きさやハルシネーションと呼ばれる不正確な生成といった問題も顕在化している。これに対し、小規模言語モデル（SLM）は効率性と安定性の観点から再評価されつつあるが、その性能は一般に LLM に劣ると理解されている。

しかし、この理解は「モデルサイズが性能を決定する」という暗黙の前提に依存している。本稿ではこの前提を問い直し、言語モデルの性能はモデルの大きさではなく、「解空間の構造」と「制約の設計」によって規定されるという立場を取る。

言語モデルは本質的に、入力 x に対して出力 y を生成する確率的過程

$$y \sim p_\theta(y \mid x)$$

として定式化される。このとき、通常はすべての可能な出力からなる空間 $\mathcal{Y}$ が暗黙に前提とされる。しかし実際の問題において意味を持つ解は、そのような広大な空間の中のごく一部に限られる。

この事実を明示するため、本稿では制約集合 C を導入し、許容される解空間を

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

として定義する。ここで重要なのは、問題の本質が $\mathcal{Y}$ の探索ではなく、$\mathcal{Y}_C$ の探索にあるという点である。

この視点に立つと、モデルに必要な能力は

$$\text{Capacity}_{required} \propto \dim(\mathcal{Y}_C)$$

によって決まる。すなわち、解空間が制約により十分に圧縮される場合、小規模なモデルでも高い性能を達成できる。

さらに、$\mathcal{Y}_C$ は一様な集合ではなく、決定的部分と曖昧な部分に分解される。このとき曖昧性は単なる不確実性ではなく、制約内に残された探索余地として理解されるべきである。本稿ではこの曖昧性を設計対象として捉え、その局所化および制御の重要性を論じる。

また、Transformer はこの枠組みにおいて「制約を仮定しない探索機構」として位置付けられる。すなわち、LLM は制約が弱い状況における最大自由度の探索器であり、SLM は制約によって圧縮された空間に適応した探索器である。

以上の観点から、本稿の目的は次の問いに集約される：

どのように解空間を制約によって構成し、その中で曖昧性を設計するか

この問いは単なるモデル選択の問題ではなく、制約と意味の設計そのものである。本稿では、解空間、制約、曖昧性、そして Transformer の性質を統一的に扱うことで、SLM と LLM を対立的にではなく、構造的に理解する枠組みを提示する。

最終的に示されるのは、次の主張である。すなわち、モデルの性能はモデルサイズではなく、解空間の設計によって決まる。そして意味とは、制約によって安定化された構造に他ならない。

大規模言語モデル（LLM）は高い表現力を持つ一方で、計算資源の大きさやハルシネーションの問題を伴う。他方、小規模言語モデル（SLM）は軽量で安定的であるが、一般には能力不足と見なされることが多い。

しかし、ドメインが限定され、適切な制約が設計される場合、この評価は逆転しうる。本稿では、制約・曖昧性・Transformer の性質という観点から、SLM と LLM の関係を再解釈する。

2　モデルと解空間

本稿の議論は、言語モデルの性能を「モデルサイズ」ではなく「解空間の構造」として捉え直すことに基づく。本節ではその基礎となる枠組みを定義し、後続の制約および曖昧性の議論への橋渡しを行う。

入力を $x \in \mathcal{X}$、出力を $y \in \mathcal{Y}$ とする。言語モデルは通常、条件付き分布

$$y \sim p_\theta(y \mid x)$$

を近似し、その最大尤度解

$$y = \arg\max_{y' \in \mathcal{Y}} p_\theta(y' \mid x)$$

を求める。このとき $\mathcal{Y}$ は理論的には極めて大きな空間であり、すべての可能な列、構造、意味表現を含む。

しかし実際の問題においては、すべての $y \in \mathcal{Y}$ が同等に意味を持つわけではない。多くのドメインでは、出力は特定の構造や規則に従う必要があり、その結果、実質的な解空間は部分集合へと制限される。

この制限を明示的に表すため、制約集合 C を導入し、許容解空間を

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

と定義する。

ここで重要なのは、モデルが本来探索すべき空間は $\mathcal{Y}$ ではなく $\mathcal{Y}_C$ であるという点である。すなわち、問題は

$$y = \arg\max_{y' \in \mathcal{Y}_C} p_\theta(y' \mid x)$$

として定式化されるべきである。

このとき、モデルの必要能力は単純なパラメータ数ではなく、探索すべき空間の大きさに依存する：

$$\text{Capacity}_{required} \propto \dim(\mathcal{Y}_C)$$

したがって、制約が弱く

$$\mathcal{Y}_C \approx \mathcal{Y}$$

である場合には広大な探索能力が必要となり、LLM が有効となる。一方、制約により

$$\mathcal{Y}_C \ll \mathcal{Y}$$

が成立する場合、必要な能力は大幅に低下し、SLM でも十分な性能が達成可能となる。

さらに、$\mathcal{Y}_C$ は一様な集合ではなく、内部構造を持つ。多くの場合、解は決定的部分と非決定的部分に分解できる：

$$y = (y_{det}, y_{amb})$$

ここで y_{det} は制約によってほぼ一意に決まる部分、y_{amb} は複数の候補を許す部分である。この分解により、解空間は

$$\mathcal{Y}_C = \mathcal{Y}_{det} \times \mathcal{Y}_{amb}$$

と表現される。

この構造は次節以降で扱う曖昧性の議論の基礎となる。すなわち、曖昧性とは $\mathcal{Y}_{amb}$ の大きさや構造として理解されるべきであり、単なるノイズや誤差ではない。

また、この分解は計算構造にも直接影響する。元の問題が

$$\mathcal{Y} \text{ 上の探索}$$

であったものが、

$$\mathcal{Y}_C \text{ 上の制約付き探索} \quad \rightarrow \quad \mathcal{Y}_{amb} \text{ 上の局所探索}$$

へと変換される。

この変換こそが、後に述べる「制約による問題変換」の核心である。すなわち、問題の本質は生成から選択、あるいは低次元探索へと移行する。

以上の議論から、言語モデルを次のように再定義できる：

$$\text{言語モデル} = \text{解空間上の確率的探索器}$$

そしてその性能は、

$$\text{モデルの大きさ ではなく 解空間の構造と制約}$$

によって規定される。

この視点に立つと、以降の議論は自然に導かれる。制約は $\mathcal{Y}$ を $\mathcal{Y}_C$ に縮退させ、曖昧性は $\mathcal{Y}_{amb}$ として残り、Transformer はその探索機構として機能し、SLM と LLM の違いはこの空間に対する適合性として理解される。

したがって本稿における中心問題は、次のように要約される：

$$\text{どのように } \mathcal{Y}_C \text{ を設計し、その中の探索を制御するか}$$

これは単なるモデル選択の問題ではなく、制約と意味の設計そのものである。

言語モデルは入力 x に対して出力 y を生成する確率分布

$$y \sim p_\theta(y \mid x)$$

を近似する。

ここで重要なのは、モデルの性能は単なるパラメータ数ではなく、「探索すべき解空間」に依存するという点である。制約集合 C を導入すると、解空間は

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

に縮退する。

このとき必要なモデル能力は概念的に

$$\text{Capacity} \propto \dim(\mathcal{Y}_C)$$

と考えられる。したがって、制約によって $\mathcal{Y}_C$ を小さくできれば、SLM であっても十分な性能を発揮できる。

3 Transformer の性質と制約

Transformer は現代の LLM および多くの SLM の基盤となるモデルであり、その本質は「関係の自由生成」にある。本節では、この性質を制約および解空間の観点から再定義し、SLM との関係を明確化する。

Transformer は自己注意機構により

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

を計算する。この演算は、入力列内のすべての要素間の関係を同時に評価し、重み付けするものである。

この構造が意味するのは、Transformer が

「関係を仮定しないモデル」

であるという点である。すなわち、どの要素同士が関連するかを事前に固定せず、すべての関係を潜在的に許容する。

このとき、モデルが扱う解空間は極めて広く、

$$\mathcal{Y} \approx \text{すべての可能な列構造}$$

となる。したがって、Transformer は本質的に

最大自由度の生成モデル

である。

この性質は、制約が弱い、あるいは未知である状況において極めて有効である。すなわち、

$$C \approx \emptyset$$

あるいは

$$\mathcal{Y}_C \approx \mathcal{Y}$$

である場合、Transformer は広大な探索空間をカバーする能力を持つため、LLM としての優位性を発揮する。

しかし同時に、この自由度は制約が存在する場合には過剰となる。制約集合 C が導入されると、真の解空間は

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

へと縮退する。

ここで Transformer は依然として $\mathcal{Y}$ 全体を探索しようとするため、実際には

不要な関係の評価 + 無効解の生成

が発生する。この現象が、ハルシネーションや計算効率の低下の一因となる。

この観点から、LLM とは

制約が弱い世界における最適解

であると解釈できる。すなわち、制約を持たない、あるいは持てない状況において、最大自由度のモデルとして機能する。

一方で SLM は、この自由度を前提としない。すなわち、SLM は

$$\mathcal{Y} \to \mathcal{Y}_C$$

という変換が事前に与えられていることを前提とするモデルである。

このとき、Transformer の役割は変化する。すなわち、LLM においては

関係の発見（relation discovery）

が主目的であったのに対し、SLM では

制約内での選択（constrained selection）

へと役割が変わる。

形式的には、LLM は

$$y = \arg\max_{y' \in \mathcal{Y}} p_\theta(y' \mid x)$$

を解くのに対し、SLM は

$$y = \arg\max_{y' \in \mathcal{Y}_C} p_\theta(y' \mid x)$$

を解く。

ここで重要なのは、$\mathcal{Y}_C$ が十分に圧縮されている場合、必要なモデル容量は大幅に低下することである：

$$\text{Capacity}_{required} \propto \dim(\mathcal{Y}_C)$$

したがって、制約が強いドメインにおいては、Transformer の「全関係探索能力」は本質的ではなくなり、むしろ

過剰な自由度

として働く可能性がある。

このとき設計の焦点は、Transformer を大きくすることではなく、

どの関係を許し、どの関係を禁止するか

を明示的に定義することへと移る。

さらに、曖昧性の観点から見ると、Transformer は本来的に

$$\text{曖昧性を最大化する方向に働く}$$

モデルである。すなわち、制約がなければ

$$|\mathcal{Y}| \gg 1$$

となり、曖昧性は無制限に拡大する。

一方、制約が導入されると、

$$\mathcal{Y} \to \mathcal{Y}_C$$

となり、曖昧性は制約内に閉じ込められる。このとき Transformer は、曖昧性を生成するモデルではなく、

$$\text{局所的曖昧性を探索するモデル}$$

へと変化する。

以上より、Transformer の本質は次のように再定義できる：

$$\text{Transformer} = \text{制約がない場合の最大自由度探索器}$$

そして SLM との関係は、

$$\text{LLM} = \text{無制約下の Transformer}$$

$$\text{SLM} = \text{制約下で再解釈された Transformer}$$

として理解できる。

したがって、重要なのはモデルサイズではなく、

$$\text{制約によって Transformer の自由度をどう再構成するか}$$

である。この視点において、制約は単なる補助ではなく、Transformer の意味そのものを規定する中心的要素となる。

Transformer は自己注意機構により

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

を計算する。この構造は全トークン間の関係を一様に扱うため、関係が未知な問題では強力である。

しかし、構造が既に存在する場合、この自由度は過剰となる。例えば分子グラフ（ZINC データセット）においては、意味は主に局所的な結合構造に依存する。この場合、近傍集約を前提とする GNN は強い帰納バイアスを持つ一方、Transformer は不要な長距離関係まで考慮してしまう。

実験的にも、GIN と Graph Transformer を比較すると、精度はほぼ同等か、むしろ GNN がわずかに優位である一方、Transformer は GPU 使用率のみが上昇する傾向が観察される。これは

$$\text{計算量の増加} \neq \text{意味の増加}$$

であることを示している。

4　制約による問題変換

制約は単なる出力の制限ではなく、問題そのものの計算構造を変換する作用を持つ。本節では、この変換が特に SLM（Small Language Model）にとってどのような意味を持つかを、ドメイン構造と対応づけて定式化する。

まず、通常の生成問題は

$$y = \arg\max_{y' \in \mathcal{Y}} p_\theta(y' \mid x)$$

と書ける。このときモデルは広大な解空間 $\mathcal{Y}$ を探索する必要があり、そのためには大きなモデル容量が要求される。

ここで制約集合 C を導入すると、問題は

$$y = \arg\max_{y' \in \mathcal{Y}_C} p_\theta(y' \mid x)$$

へと変換される。ただし

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid \forall c_i \in C,\ c_i(y, x) = 1\}$$

である。

このとき重要なのは、制約が「探索範囲の削減」ではなく、「問題の型変換」を引き起こす点である。すなわち、生成問題は次のように再構成される：

$$\mathcal{Y} \longrightarrow \mathcal{Y}_C \longrightarrow \text{構造化空間} \longrightarrow \text{有限候補集合}$$

特に $\mathcal{Y}_C$ が有限集合または低次元構造に収束する場合、

$$y \in \{y_1, y_2, \ldots, y_k\}$$

となり、問題は生成ではなく選択問題へと変換される。このとき必要なモデル能力は

$$\text{生成能力} \ \rightarrow\ \text{識別能力}$$

へとシフトする。

この変換こそが、SLM を有効にする核心である。すなわち、SLM は広大な生成空間を探索する能力は持たないが、制約により圧縮された空間内での選択や局所探索には十分対応できる。

この観点をドメインごとに具体化する。

まず、SQL 生成のドメインでは、スキーマ（テーブル構造、カラム型、リレーション）が強い制約として機能する。このとき出力空間は

$$\mathcal{Y}_C \subset \text{構文的に正しい SQL 集合}$$

に制限されるだけでなく、実質的には特定のスキーマに整合するクエリ集合に縮退する。結果として問題は

$$\text{自然言語} \rightarrow \text{有限個の SQL 候補の選択}$$

へと変換される。

次にコード生成では、文法、型、API 仕様が制約として作用する。このとき

$$\mathcal{Y}_C \subset \text{コンパイル可能コード}$$

となるだけでなく、問題は特定の関数仕様に対する実装候補の選択へと還元される。この場合、SLM でも十分な精度が得られる。

法律文書の解析では、条文構造、定義語、参照関係が制約となる。このとき出力は

$$\mathcal{Y}_C \subset \text{条文構造に整合する意味表現}$$

に制限され、曖昧性は限定された解釈部分にのみ残る。ここでも問題は部分的選択問題へと変換される。

さらに、グラフデータ（例えば分子構造やエピトープ推定）においては、ノード間の接続関係や距離制約が強く作用する。この場合、

$$\mathcal{Y}_C \subset \text{物理的・化学的に実現可能な構造}$$

となり、探索空間は劇的に縮小される。特にエピトープ推定では、表面性や距離クラスタといった制約により、候補は空間的に局所化される。

このように、ドメインが持つ構造を制約として明示化することで、問題は

$$\text{非構造な生成問題} \ \rightarrow \ \text{構造化選択問題}$$

へと変換される。

ここで重要なのは、制約がモデル外に存在しうる点である。すなわち、

$$\text{モデル能力} + \text{外部制約} \ \approx \ \text{全体性能}$$

と捉えるべきであり、制約はモデルの一部ではなく、計算系全体の構成要素である。

この視点から見ると、LLM は制約が弱い場合に有効であり、SLM は制約が強い場合に有効である。より正確には、

$$\text{Capacity}_{required} \propto \dim(\mathcal{Y}_C)$$

であるため、$\mathcal{Y}_C$ を十分に圧縮できる場合、SLM で十分となる。

最後に、この変換の設計原理をまとめる。すなわち、ドメイン構造を抽出し、それを制約として形式化し、解空間を圧縮し、生成問題を選択問題へと変換することである。このとき、モデルの役割は「意味生成」から「構造内選択」へと変化する。

したがって、SLM を有効に使うための本質は、

$$\text{制約によって問題をどこまで変換できるか}$$

にある。

制約は単なる制限ではなく、問題の性質を変換する操作である。本来の生成問題

$$y = \arg\max_{y'} p(y' \mid x)$$

は、制約により

$$y = \arg\max_{y' \in \mathcal{Y}_C} p(y' \mid x)$$

へと変換される。

さらに $\mathcal{Y}_C$ が有限集合に近づくと、

$$y \in \{y_1, y_2, \ldots, y_k\}$$

という選択問題に変わる。

この変換は実務上極めて重要である。例えば SQL 生成ではスキーマ制約により出力は有限の構造に限定される。同様にコード生成では文法と型が強い制約として機能する。このとき SLM でも高い精度が達成される。

5 曖昧性の再定義

曖昧性は従来、除去すべき不確実性と考えられてきた。しかし本稿ではこれを

探索余地（exploration space）

として再定義する。

制約集合 C により定義される解空間を

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

とする。このとき、曖昧性とは

$$|\mathcal{Y}_C| > 1$$

である状態、すなわち複数の正当な解が存在する状況を指す。

完全に制約された場合、

$$|\mathcal{Y}_C| = 1$$

となり、探索は存在しない。この系は安定であるが、表現の自由度や創造性を持たない。

一方で、

$$|\mathcal{Y}_C| > 1$$

であれば、複数の解が許容される。このとき曖昧性は単なる不確実性ではなく、

多様な解を生成するための構造的自由度

となる。

ここで曖昧性は情報理論的には、条件付き分布 $p(y \mid x, C)$ のエントロピーとして捉えることもできる：

$$H(Y \mid x, C) = -\sum_{y \in \mathcal{Y}_C} p(y \mid x, C) \log p(y \mid x, C)$$

この値が高いほど、曖昧性は大きい。

ただし重要なのは、曖昧性の大きさそのものではなく、その構造である。すなわち、

$$\mathcal{Y}_C = \mathcal{Y}_{det} \cup \mathcal{Y}_{amb}$$

と分解できるとき、曖昧性は $\mathcal{Y}_{amb}$ にのみ存在する。この分解が可能である場合、曖昧性は制御可能となる。

さらに、曖昧性は制約と対立するものではなく、制約によって定義されるものである。制約が存在しなければ、

$$\mathcal{Y}_C \approx \mathcal{Y}$$

となり、曖昧性は無制限に拡散し、意味を持たない。

したがって曖昧性とは、

制約によって囲い込まれた探索領域

と理解すべきである。

この観点から見ると、曖昧性は排除すべき対象ではなく、設計対象である。制約を強くすれば曖昧性は消失し、制約を緩めれば曖昧性は拡大する。この調整により、安定性と創造性のバランスが決定される。

最終的に、曖昧性は次のように位置付けられる：

$$\text{曖昧性} = \text{制約内に存在する可変性}$$

これは単なる不確実性ではなく、意味生成に必要な最小限の自由度である。

曖昧性は従来、除去すべき不確実性と考えられてきた。しかし本稿ではこれを

探索余地（exploration space）

として再定義する。

完全に制約された場合、

$$|\mathcal{Y}_C| = 1$$

となり、探索は存在しない。一方、

$$|\mathcal{Y}_C| > 1$$

であれば、複数の解が存在し、探索が可能になる。

例えば文章生成において、トーンや表現スタイルを複数候補として残す設計を行うと、曖昧性は創造性の源として機能する。

6　曖昧性の局所化

曖昧性は排除すべき対象ではなく、設計すべき対象である。本節では、曖昧性をどのように局所化し、制御可能な形で保持するかを形式的に記述し、具体例を通じてその有効性を示す。

まず、制約集合 C により定義される許容解空間を

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

とする。一般に $\mathcal{Y}_C$ は一様ではなく、決定的部分と曖昧部分に分解できる：

$$\mathcal{Y}_C = \mathcal{Y}_{det} \cup \mathcal{Y}_{amb}, \quad \mathcal{Y}_{det} \cap \mathcal{Y}_{amb} = \emptyset$$

ここで $\mathcal{Y}_{det}$ は一意に決まる部分、$\mathcal{Y}_{amb}$ は複数の候補を許す部分である。

曖昧性の局所化とは、$\mathcal{Y}_{amb}$ の自由度を制御可能な部分空間へと押し込める操作であり、形式的には

$$\dim(\mathcal{Y}_{amb}) \ll \dim(\mathcal{Y}_C)$$

を満たすように設計することを意味する。

このとき、出力構造を

$$y = (y_{det}, y_{amb})$$

と分解でき、y_{det} は制約により固定され、y_{amb} のみが探索対象となる。したがって生成問題は

$$y_{amb} = \arg\max_{y' \in \mathcal{Y}_{amb}} p_\theta(y' \mid x, y_{det})$$

へと還元される。

この構造は、モデルが扱う自由度を大幅に削減しつつ、必要な創造性のみを保持する。

具体例として、まず自然言語生成を考える。文章生成において「主語・述語・時制」は文法制約によりほぼ決定される一方、「語彙選択」や「文体」は複数の選択肢を持つ。この場合、

$$y_{det} = \{\text{構文構造}\}, \quad y_{amb} = \{\text{語彙・表現}\}$$

となり、曖昧性は語彙レベルに局所化される。この設計により、文法破綻を防ぎつつ表現の多様性を維持できる。

次にコード生成の例では、型・構文・API 仕様は厳密に決定される一方、アルゴリズム選択（ループか再帰かなど）は複数の正解が存在する。このとき

$$y_{det} = \{\text{型・構文・API 制約}\}, \quad y_{amb} = \{\text{実装戦略}\}$$

となる。ここでも曖昧性は限定された設計空間に閉じ込められる。

法律文書解析においては、条文番号や数値条件は決定的である一方、「合理的な範囲」や「相当な理由」といった部分が曖昧性を持つ。この場合、

$$y_{det} = \{\text{条文構造・数値条件}\}, \quad y_{amb} = \{\text{解釈部分}\}$$

となる。曖昧性は意味解釈のみに局所化される。

グラフ構造の例では、ノード集合とエッジの存在は制約により固定される一方、ノードの埋め込み表現やクラスタリング結果は複数の等価解を持つ。このとき

$$y_{det} = \{\text{グラフ構造}\}, \quad y_{amb} = \{\text{埋め込み・クラスタ}\}$$

となる。特に GNN は y_{det} を強く仮定することで効率を得ている。

エピトープ推定においても同様である。タンパク質表面という幾何学的制約や接触距離制約は決定的である一方、どの残基集合が最も重要かは複数候補が存在する。このとき

$$y_{det} = \{\text{表面・距離制約}\}, \quad y_{amb} = \{\text{残基集合の選択}\}$$

となり、曖昧性はクラスタ選択に局所化される。

ここで重要なのは、曖昧性を局所化しない場合の挙動である。もし構造全体に曖昧性が広がると、

$$\mathcal{Y}_{amb} \approx \mathcal{Y}_C$$

となり、モデルは広大な探索空間を扱う必要がある。このとき SLM では能力不足となり、LLM であってもハルシネーションが増加する。

逆に、曖昧性が局所化されている場合、探索は低次元空間に限定されるため、

$$\text{探索コスト} \propto \dim(\mathcal{Y}_{amb})$$

が小さくなり、SLM でも十分に扱える。

さらに、曖昧性の局所化は安全性とも密接に関係する。ハルシネーションは

$$y \notin \mathcal{Y}_C$$

として定義されるが、$\mathcal{Y}_{det}$ が強く固定されている場合、モデルが制約外へ逸脱する余地は著しく減少する。

したがって、設計原則は次のようにまとめられる。すなわち、構造の大部分を決定的に固定し、曖昧性を低次元かつ意味的に許容された部分空間に閉じ込めることである。

このとき、モデルは

$$\text{固定構造の上での局所探索}$$

として振る舞い、安定性と創造性を同時に達成する。この構造こそが、制約駆動アーキテクチャにおける中核的設計原理である。

重要なのは曖昧性を排除することではなく、その存在を局所化することである。構造を

$$\mathcal{S} = \mathcal{S}_{fixed} \times \mathcal{S}_{amb}$$

と分解すると、設計の目的は $\mathcal{S}_{amb}$ を限定された部分に閉じ込めることである。

例えば契約書解析において

$$\{\text{termination}, \text{notice\_days}, \text{conditions}\}$$

のようにスロット化すると、曖昧性は「conditions」のみに集中する。このようにして曖昧性の影響範囲を制御できる。

7　曖昧性とハルシネーションの境界

曖昧性とハルシネーションはしばしば混同されるが、本質的には異なる概念である。本節ではその違いを形式的に定義し、さらに具体例を通じてその境界を明確化する。

まず、制約集合 C により定義される許容解空間を

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

とする。

このとき、曖昧性は

$$|\mathcal{Y}_C| > 1 \quad \text{かつ} \quad y \in \mathcal{Y}_C$$

である状態であり、複数の正当な解が存在する状況を指す。一方、ハルシネーションは

$$y \notin \mathcal{Y}_C$$

であり、制約違反による不正な生成である。

この区別は単なる定義にとどまらず、実務的にも極めて重要である。以下に具体例を示す。

まず自然言語生成の例として、翻訳タスクを考える。英語文「He saw her duck」は文脈により「彼は彼女のアヒルを見た」あるいは「彼は彼女が身をかがめるのを見た」と解釈できる。この場合、両者は文法的・意味的制約を満たしており、

$$y_1, y_2 \in \mathcal{Y}_C$$

であるため曖昧性である。しかし、「彼は彼女の飛行機を見た」という訳が生成された場合、それは入力と整合せず、

$$y \notin \mathcal{Y}_C$$

となるためハルシネーションである。

次に法律文書の例を考える。契約書に「30 日前の書面通知により解約可能」とある場合、「営業日換算か暦日換算か」は文脈によって複数の解釈があり得る。このような場合は曖昧性である。一方、モデルが「60 日前の通知が必要」と回答した場合、それは条文に存在しない内容であり、ハルシネーションである。

コード生成においても同様である。ある仕様に対して複数の実装（例えばループ構造と再帰構造）が存在する場合、いずれも正しい実装であり曖昧性に属する。しかし、存在しない API や未定義関数を呼び出すコードを生成した場合、それはハルシネーションである。

グラフ構造の例も興味深い。分子グラフにおいて、同一の分子を表す異なるグラフ表現（例えばノード順序の違い）はすべて制約を満たすため曖昧性である。一方、存在しない結合を追加したり、化学的に不可能な構造を生成した場合、それは制約違反でありハルシネーションとなる。

さらにエピトープ推定の文脈では、複数の候補残基集合が空間的クラスタとして成立しうる場合、それらはすべて

$$\mathcal{Y}_C$$

に属する曖昧な解である。しかし、タンパク質表面に存在しない内部残基をエピトープとして選択した場合、それは物理的制約を満たさず、ハルシネーションである。

ここで重要な観点として、「制約の明示性」が挙げられる。制約が弱い、あるいは明示されていない場合、

$$\mathcal{Y}_C \approx \mathcal{Y}$$

となり、曖昧性とハルシネーションの区別が曖昧になる。このとき、ハルシネーションは「それっぽさ」として観測される。

逆に制約が強く定義されると、

$$\mathcal{Y}_C \ll \mathcal{Y}$$

となり、両者の境界は明確になる。

したがって設計上の本質は、

$$\text{制約により } \mathcal{Y}_C \text{ を明確に定義すること}$$

にある。

さらに一歩進めると、曖昧性は単なる副作用ではなく、設計可能な対象である。すなわち、

$$\mathcal{Y}_C = \mathcal{Y}_{det} \cup \mathcal{Y}_{amb}$$

と分解し、$\mathcal{Y}_{amb}$ を意図的に設計することで、創造性や多様性を制御できる。

最終的に、曖昧性とハルシネーションの境界は次の一行に集約される：

$$\text{曖昧性} = \text{制約内の多様性}, \quad \text{ハルシネーション} = \text{制約外の生成}$$

この区別を明確に扱うことが、SLM 設計および制約駆動アーキテクチャの核心である。

曖昧性とハルシネーションは本質的に異なる。

曖昧性は

$$y \in \mathcal{Y}_C \quad \text{かつ} \quad |\mathcal{Y}_C| > 1$$

である状態であり、すべての候補が制約を満たす。

一方、ハルシネーションは

$$y \notin \mathcal{Y}_C$$

であり、制約違反である。

例えば法律文書において、条文内で複数の解釈が存在する場合は曖昧性であるが、存在しない条文を生成した場合はハルシネーションである。

したがって両者の境界は

$$y \in \mathcal{Y}_C \text{ か否か}$$

によって明確に定義できる。

8 SLM が有効となる条件

SLM（Small Language Model）が LLM に匹敵、あるいはそれ以上の有効性を持つか否かは、モデルサイズそのものではなく、問題が持つ構造と制約の設計に依存する。本節ではその条件を、ドメイン構造、制約設計、曖昧性の取り扱いという三つの観点から定式化する。

まず、問題領域（ドメイン）を $\mathcal{D}$ とし、入力を $x \in \mathcal{X}$、出力を $y \in \mathcal{Y}$ とする。通常の生成問題は

$$y \sim p_\theta(y \mid x), \quad y \in \mathcal{Y}$$

であるが、実際のドメインでは $\mathcal{Y}$ は均一な空間ではなく、内部に構造 $\mathcal{S}$ を持つ。この構造はしばしば

$$\mathcal{S} = (\mathcal{E}, \mathcal{R})$$

と書ける。ここで $\mathcal{E}$ は要素集合（例：トークン、ノード、スロット）、$\mathcal{R}$ はそれらの関係である。

SLM が有効となる第一の条件は、この構造 $\mathcal{S}$ が明示的あるいは暗黙的に安定していることである。すなわち、ドメインが次の性質を持つときである：

$$\mathcal{S}(x) \approx \mathcal{S}(x') \quad (\forall x, x' \in \mathcal{D})$$

これは「問題ごとに構造が大きく変わらない」ことを意味する。例えば SQL 生成ではスキーマ構造、分子予測では結合グラフ構造、法律文書では条文構造がこれに該当する。

次に、制約 C を導入する。制約とは、出力が満たすべき条件の集合であり、

$$C = \{c_i\}_{i=1}^{m}$$

と表される。このとき許容される解空間は

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid \forall c_i,\ c_i(y, x) = 1\}$$

となる。

SLM が有効となる第二の条件は、この $\mathcal{Y}_C$ が十分に圧縮されることである。すなわち、

$$\frac{|\mathcal{Y}_C|}{|\mathcal{Y}|} \ll 1$$

が成立する場合である。これは「問題が本質的に制約充足問題へ変換できる」ことを意味する。

ここで重要なのは、制約が単なるフィルタではなく、生成過程そのものを変形する点である。生成は

$$y = \arg\max_{y'} p_\theta(y' \mid x)$$

ではなく、

$$y = \arg\max_{y' \in \mathcal{Y}_C} p_\theta(y' \mid x)$$

へと変換される。極端な場合、$\mathcal{Y}_C$ が有限集合に収束すると、

$$y \in \{y_1, y_2, \ldots, y_k\}$$

となり、問題は生成ではなく選択となる。このとき必要なモデル能力は大幅に低下する。

第三の条件は、曖昧性の構造が制御可能であることである。曖昧性は完全に排除されるべきではなく、むしろ次のように分解される：

$$\mathcal{Y}_C = \mathcal{Y}_{fixed} \cup \mathcal{Y}_{amb}$$

ここで $\mathcal{Y}_{fixed}$ は一意に決まる部分、$\mathcal{Y}_{amb}$ は複数候補を許す部分である。

重要なのは、この曖昧部分が局所化されていることである。すなわち、

$$\dim(\mathcal{Y}_{amb}) \ll \dim(\mathcal{Y}_C)$$

である場合、曖昧性は制御可能となる。

例えば契約書解析において、日付や条文番号は一意に決まるが、条件文の解釈のみが曖昧である。この場合、曖昧性は限定されたサブスペースに閉じ込められている。

ここで曖昧性とハルシネーションの区別が重要となる。曖昧性は

$$y \in \mathcal{Y}_C$$

を満たす複数解の存在であり、すべて制約内にある。一方ハルシネーションは

$$y \notin \mathcal{Y}_C$$

であり、制約違反である。

したがって設計上の目標は、

$$\mathcal{Y}_{amb} \subset \mathcal{Y}_C$$

を維持しつつ、$\mathcal{Y}_C$ の外部への逸脱を防ぐことである。

以上をまとめると、SLM が有効となるのは以下の条件が同時に成立する場合である。ドメインが安定した構造を持ち、その構造が制約として表現可能であり、その制約によって解空間が圧縮され、さらに曖昧性が局所化されている場合である。

このとき、問題の本質は「生成能力」ではなく、

$$\text{構造} + \text{制約} + \text{局所的探索}$$

となる。SLM はこの三者を満たす範囲において、LLM に匹敵するだけでなく、むしろ過剰な自由度を持たないという点で安定した解を与える。

9 結論

本稿では、言語モデルの性能を「モデルサイズ」ではなく「解空間の構造と制約」によって捉え直した。この視点に立つと、LLM と SLM の関係は単なる能力差ではなく、異なる問題設定への適合として理解される。

言語モデルは本質的に、入力 x に対して出力 y を生成する確率的探索器であり、

$$y = \arg\max_{y' \in \mathcal{Y}} p_\theta(y' \mid x)$$

を解く。しかし実際の問題では、意味を持つ解は制約集合 C によって規定され、

$$\mathcal{Y}_C = \{y \in \mathcal{Y} \mid y \text{ satisfies } C\}$$

へと縮退する。

したがって、問題の本質は $\mathcal{Y}$ の探索ではなく、

$$\mathcal{Y}_C \text{ 上の探索}$$

である。

このとき、モデルに要求される能力は

$$\text{Capacity}_{required} \propto \dim(\mathcal{Y}_C)$$

によって決まり、解空間が十分に圧縮される場合、SLM であっても LLM に匹敵する性能が達成される。

さらに、$\mathcal{Y}_C$ は一様ではなく、

$$\mathcal{Y}_C = \mathcal{Y}_{det} \times \mathcal{Y}_{amb}$$

と分解される。ここで $\mathcal{Y}_{det}$ は制約によって固定される部分、$\mathcal{Y}_{amb}$ は曖昧性として残る部分である。

曖昧性は従来、不確実性として扱われてきたが、本稿ではこれを

$$\text{制約内に存在する探索余地}$$

として再定義した。したがって、曖昧性は排除すべき対象ではなく、設計すべき対象である。

このとき重要なのは、曖昧性とハルシネーションの区別である。曖昧性は

$$y \in \mathcal{Y}_C$$

を満たす複数解の存在であり、ハルシネーションは

$$y \notin \mathcal{Y}_C$$

である。この区別は、制約の明示性によって初めて成立する。

Transformer はこの枠組みにおいて、「制約を仮定しない最大自由度の探索器」として理解される。すなわち、

$\mathcal{Y}$ 全体を対象とする探索機構

である。このため、制約が弱い状況では LLM として有効に機能するが、制約が強い状況では過剰な自由度を持つ。

一方、SLM は

$$\mathcal{Y} \to \mathcal{Y}_C$$

という変換が前提とされる環境において有効であり、その役割は

制約内での局所探索

にある。

以上より、モデル設計の本質は次の問いに帰着する：

どの制約によって $\mathcal{Y}$ を $\mathcal{Y}_C$ に変換するか

どこに曖昧性 $\mathcal{Y}_{amb}$ を残すか

この問いは、単なる機械学習の問題ではなく、制約と意味の設計そのものである。すなわち、

意味とは、制約によって安定化された構造である

したがって、本稿の結論は次の一点に集約される。モデルの性能はモデルの大きさではなく、解空間の設計によって決まる。制約によって解空間を構成し、曖昧性を適切に局所化することで、小規模モデルでも高い性能と安定性を両立できる。

この視点において、LLM と SLM の対立は消失する。それらは単に、異なる制約構造に適応した探索器にすぎない。問題はモデルではなく、どのような世界（解空間）を構成するかにある。

本稿の主張は、言語モデルの性能はモデルサイズではなく、制約と解空間の設計によって規定されるという点にある。適切な制約により問題を選択問題へと変換すれば、小規模モデルでも高い性能を発揮できる。

さらに、曖昧性は排除すべき対象ではなく、設計対象である。曖昧性を局所化し、制約内に閉じ込めることで、安定性と創造性を同時に達成できる。

最終的に、この問題は次の問いに帰着する：

どの構造を固定し、どこに曖昧性を残すか

これは単なるモデル選択の問題ではなく、制約と意味の設計そのものである。