

知識と LLM

松田裕幸（シンボリックシステムズ）

知識をなんらかの言語で書かれたものと規定すると、自然言語によるものは当然として、プログラミング言語を使い記述されたプログラムコードも知識とみなすことができる。ここでは、そうした知識と LLM (Large Language Model) との関係を考察する。

1. 閉じた体系／開いた世界.....	45
1.1 プログラミング言語 PL	46
1.2 形式的アプローチの自然言語への拡張	46
1.3 LLM (>Transformer)	46
2. LLM における知識の理解.....	47
2.1 パターン主導理解	47
2.2 逆関数の理解	48
2.3 指示の理解	48
2.4 物理制約の理解.....	49
2.5 推論	49
3. LLM における知識の学習	52
3.1 transformer の基本構造	52
3.2 埋め込み embedding と自己注意 self-attention	52
4. LLM と知識の活用	53
4.1 AI ロボットへの指示の埋め込み	53
4.2 RAG(Retrieval-Augmented Generation).....	53

1. 閉じた体系／開いた世界

はじめに形式言語について以下の点を確認しておく。

- 構文規則 syntax と意味規則 semantics が明確に定義される。
- 意味体系が構文要素に関し、**閉じている**。
- 構文要素に対し一意の意味が割り当てられる。

ついで形式言語の例としてプログラミング言語を取り上げ、そこでの構文体系と意味体系を議論し、同じ議論の延長線上で自然言語を扱い、その限界を超えるものとして transformer あるいはその進化版である LLM を導入する。

1.1 プログラミング言語 PL

プログラミング言語 PL によって記述されたプログラムはチョムスキーの文脈自由文法 CFG (の部分集合) によって受理される。そしてその意味は言語 PL が規定する意味の体系に準じて定まる。

プログラム_{inPL} ∈ 文脈自由文法 CFG の部分集合
意味(プログラム_{inPL}) ∈ 意味体系(PL)

意味(プログラム_{inPL}) は実際、次のような意味解釈行為によって明確に (一意に) 定義される。

- Compiler の場合: 意味解釈の結果として各意味に機械語コードを割り当てる
- Interpreter の場合: 意味解釈の結果としてただちにプログラムコードを実行する

1.2 形式的アプローチの自然言語への拡張

形式的アプローチを自然言語定義に拡張したものとしてモンタギュー文法や拡張句構造文法 HPSG (Head-driven Phrase Structure Grammar) が知られており、対象とする自然言語クラスに関し、構文と意味は閉じている

テキスト_{inHPSG} ∈ HPSG
意味(テキスト_{inHPSG}) ∈ 意味体系(HPSG)

が、あらゆる自然言語のクラスを扱うことはできない:

HPSG ⊂ 自然言語 NL

1.3 LLM (>Transformer)

LLM が登場して、はじめて「**有限**の世界の言葉で (基本) **無限**の世界を表現」することができるようになった。次式

テキスト_{inNL} ∈ L(LLM)
意味(テキスト_{inNL}) ∈ コーパス(LLM)

において、左辺の集合は無限 (開 open) なのに対し、右辺は学習量において有限 (閉 close) に抑えられている。またテキスト構成要素をトークン token としたとき、トークンと意味の関係は 1 : n (つまり多義) であり、トークン文脈とコーパスに関する統計的出現確率によって 1 : 1 (つまり一義) に抑えられる。

なお、形式言語における意味は意味規則によって演繹的に厳密に規定される。これに反し、（膨大な事例からなる）コーパスは帰納的に意味を規定する。開放系である自然言語を対象とした場合、形式言語理論が保証する厳密さを失っても、膨大な事例が担保する（！まさに帰納的）一定程度の品質によって LLM の存在意義は大きくなっていくと予想される。

[挿話： コーパス corpus]

コロケーション collocation の集合をコーパスと呼ぶ。

コロケーション的に間違った例。

「ビタミンを x 食べる（○摂取する）」

「雨が降ってきたので傘を x 使った（○さした）」

「近年、コンビニ店舗は x すごい（○著しい）増加傾向にある」

単語「ゆっくり」に対応するコロケーションとコーパスにおける出現頻度の例

コロケーション	頻度
<u>ゆっくり</u> 休む	119
<u>ゆっくり</u> する	114
<u>ゆっくり</u> 歩く	118
<u>ゆっくり</u> 寝る	62
...	...

2. LLM における知識の理解

2.1 パターン主導理解

これはバブルソートの Python 版である。変数名も一般的によく使う名前 i, j, L 等ではなく、あえて abc, xyz, Y のように見慣れない名前に変更して ChatGPT4 に「何をするプログラム」かを尋ねると「並び替えを行うプログラムであり、アルゴリズムはバブルソート」だと正解する。しかも $Y[xyz - 1] = Y[xyy]$ において「 xyy が誤り」であるという指摘してくれる。

```
for abc in range(len(Y)):  
    for xyz in range(len(Y)-1, abc, -1):
```

```

if Y[xyz-1] > Y[xyz]:
    x = Y[xyz-1]; Y[xyz-1] = Y[xyy]; Y[xyz] = x

```

この手品のような結果を導出する仕掛けは何にあるのだろうか。それはパターンである。上記コードのパターンを論理構造として書き下すと次のようになる。

```

∀ i in 0..length(L)
  ∀ j in length(L)-1..i
    swap(L[j-1], L[j]) if L[j-1]>L[j]

```

LLM は膨大なコードサンプルとそれに付随するコメントあるいはドキュメントを学習することでこのようなコードパターンとバブルソートアルゴリズム結びつける。ただしそこに明示的な理解「バブルソートアルゴリズムは。。。の形をしている」はなく、あくまで事例とターム（ここではバブルソート）を関連づけているにすぎない。なおパターンという表現は transformer にも LLM にも登場しない。筆者独自の使用である。

パターンという点ではコロケーションもパターンを形成する、と本稿ではとらえている。また特定のコードが「ある」パターンに合致するかどうかは一意に定まらず学習量に応じた蓋然性（統計的確率）に依存する。

2.2 逆関数の理解

「りんご」から連想されるものは多数ある。連想関数 $f(\text{apple})$ の値域として「りんご」の連想物を定義したとき、値域が apple になるような逆関数 f^{-1} を作るよう ChatGPT4 に依頼した結果が次になる。

```

f-1('秋')
f-1('知識')
f-1('おとぎ話')
f-1('ニューヨーク')
f-1('教育')

```

ここで ChatGPT4 は逆関数の定義域に「りんご」が連想するものを列挙している。これをみるかぎり、ChatGPT4 はあきらかに連想関係を理解し、かつ、連想関係の逆関数を理解しているとみなせる。ただしこれも 2.1 で触れたように膨大な学習例（「。。。は連想関係にある」「ある関数の逆関数は定義域と値域が（存在を前提として）逆転する」）から間接的に理解した結果にすぎない。

2.3 指示の理解

"There is a blue box. Pick up the box. Put this box on the table." を例にロボットへの指示を理解する。まず言語上の一致を確認する：""There is a blue box. Pick up the

box. Put this box on the table."。次にこれらを対象に意味を与える："There is a blue box. Pick up the box. Put this box on the table."。意味は具体的なロボット操作で置き換えられる。なお、「持ち上げられる」「置ける」というアフォーダンス affordance は LLM の範疇外となる。

2.4 物理制約の理解

ChatGPT4 に「500cc のビーカーにまず 300cc の水を入れました。ついで 400cc の水を追加しました。いまビーカー内の水量はいくらですか。」と尋ねると「500cc」と正解を答える。かつその理由として「 $300\text{cc} + 400\text{cc} = 700\text{cc}$ のうち 200cc はこぼれる」を挙げる。つまり 1) 簡単な計算ができ、2) 容量を超え、内容物を追加したら溢れるという常識、を使っていることがわかる。

2.5 推論

ChatGPT4 は演繹的推論 deductive inference、帰納的推論 inductive inference、アブダクション（仮説推定）abduction を操る力を持っている。以下、実際に ChatGPT4 に解かせた問題を紹介する。

(1) 演繹的推論

プロンプト：「素数が無限に存在する」ことを演繹推論を使い証明しなさい。

ChatGPT4：

1. **仮定:** 素数が有限個しか存在しないと仮定します。つまり、素数が $p_1, p_2, \dots, p_n$ の n 個の素数だけであるとします。
2. **証明の構築:** すべての素数を掛け合わせた数に 1 を加えた数を考えます。すなわち、 $P = p_1 \times p_2 \times \dots \times p_n + 1$ 。
3. **矛盾の導出:**
 - P は素数のどれにも割り切れない（割り切れた場合、その商から 1 を引いたものが元の素数の積になり、その結果は必ず 1 以外になるため、矛盾する）。
 - よって、 P は素数であるか、素数でない場合は新たな素数によってのみ割り切れる数である。いずれにしても、 P は $p_1, p_2, \dots, p_n$ 以外の新たな素数または素数の積を持つことになります。
4. **結論:** この矛盾から、素数が有限個しか存在しないという仮定が誤りであることが分かります。従って、「素数が無限に存在する」ことが証明されます。

補足。「素数が無限に存在する」に関する証明は山ほどネットに出ており、もちろん教科書にも掲載されている。ChatGPT4 はそうした知識の骨格（論理構成）を学

習しているのか、どれか一つを代表例として丸暗記しているのか、筆者は前者を支持する。

(2) 帰納的推論

今度はあえて推論方式を指定しないことにする。

プロンプト：世界各国の経済成長率と国民の幸福度スコアの関係について次のような研究（観察）結果が得られました。

- 観察 1: 高い経済成長率を示した国では、国民の幸福度スコアが時間とともに向上している。
- 観察 2: 経済成長率が低い、またはマイナス成長を記録した国では、幸福度スコアの向上は限定的または逆転している。
- 観察 3: いくつかの国では、経済成長に伴い不平等が拡大し、幸福度スコアの向上が抑えられている。

この観察結果から推論し、得られる結論を 1 つ挙げなさい

ChatGPT4: 経済成長自体が国民の幸福度向上に直接的に寄与する要素である一方で、その成果の**分配の仕方**が幸福度に対して重要な役割を果たす。

補足。ChatGPT に「この結論に至るために利用した推論方式」を尋ねたところ、誘導推論（英訳は帰納的推論と同じ Inductive Reasoning）と類推推論 Analogical reasoning を回答として返してきた。ChatGPT4 が回答した、それぞれの推論が果たした役割は次のとおりである：

- **誘導推論**：観察 1 と観察 2 から、「経済成長率が高い国では国民の幸福度が向上する傾向にあり、逆に経済成長率が低いかマイナス成長の国では幸福度の向上が限定的または逆転する」という一般的な傾向を導き出しました。
- **類推推論**：観察 3 から、経済成長が不平等を拡大させることが幸福度の向上を抑制するケースが存在することが示されています。この情報を基に、経済成長と国民の幸福度との関係は単純な直線的なものではなく、成長の果実の**分配の公平性**が重要であるという類推を行いました。

(3) アブダクション

プロンプト：ある研究者チームが、過去数十年にわたり特定の地域の森林面積が顕著に減少していることを発見しました。この森林は人里離れた場所にあり、直接的な人間の活動の影響を受けにくい地域です。次の 3 つの証拠を元に原因を推定できますか。推定にはアブダクションを使ってください。

- 証拠 1: 森林の減少率は、気候変動の影響が最も顕著な季節に加速する。
- 証拠 2: 近隣地域で増加している特定の害虫の種が発見された。
- 証拠 3: 森林の土壌サンプルからは、化学物質の異常な蓄積は見られなかった。

GPT：証拠の分析

- **証拠 1**：森林の減少率が気候変動の影響が最も顕著な季節に加速するという点は、この現象が自然環境の変化に密接に関連していることを示唆しています。気候変動が原因である可能性が考えられますが、直接的な因果関係を示すものではありません。
- **証拠 2**：近隣地域で増加している特定の害虫が発見されたことは、これらの害虫が森林減少に寄与している可能性を示唆しています。害虫の増加は、気候変動による生態系の変化の結果である可能性があります。
- **証拠 3**：森林の土壌サンプルから化学物質の異常な蓄積は見られなかったことは、人間による直接的な化学汚染（例えば、農薬や工業廃棄物による汚染）が森林減少の原因でないことを示しています。

原因の推定：これらの証拠をアブダクションのプロセスにより分析すると、最もらしい原因として「気候変動による生態系の変化が、特定の害虫の増加を促し、それが森林減少を加速させている」という仮説が導き出されます。気候変動は季節的な影響を強め、害虫の生存と繁殖条件を改善する可能性があります。これにより、それまで森林にとって無害だった害虫が、森林にとって脅威となり得るのです。

結論 したがって、提供された証拠から導き出される最もらしい説明は、「気候変動による影響が特定の害虫の増加を引き起こし、これが森林減少の主な原因となっている」というものです。この仮説は、土壌の化学物質蓄積の欠如という証拠によっても補強されます。なお、この仮説が正しいかどうかを確認するには、さらなる詳細な調査とデータ分析が必要です。

補足。以上3つの例を見るにつけ、ChatGPT4は3つの「概念」（演繹的推論、帰納的推論、アブダクション）に関し、その内容を丸暗記しているというより、**論理構成（推論骨格）**を学習していると結論づけて問題ないように思える。

3. LLM における知識の学習

3.1 transformer の基本構造

transformer の基本構造に入る前に、深層学習における**深層**の役割を整理しておく。

- データ⇒特徴抽出⇒分類（クラス）
- 具体⇒抽象

大量のデータが与えられたとき、層を重ねるごとに、データは有限の**クラス**に分類され、あるいは、具体的な事項を表すデータはより抽象的な**概念**に収束する。確認しておくことのもう一つが**深層学習**についてである。その過程は**順伝搬** forward propagation、**損失計算** loss（学習結果と正解の誤差）とそれに伴う**逆伝搬** back propagation に分かれ、この構成は transformer でも同様となる。そして層の前後をエンコーダ（入力変換）とデコーダ（出力変換）で挟む。

これに対し transformer では埋め込み層 embedding layer と自己注意層 self-attention layer が新たに加わる：

Embedding 層 | Encoder || {Attention 層 | 特徴抽出層}⁺ | Decoder

ここで Embedding 層は1層、Attention 層は必要に応じて何層かに分かれるが、学習の対象として（つまり学習に応じて）値が変わることは深層学習の場合と同じである。

3.2 埋め込み embedding と自己注意 self-attention

具体的な例「朝日に掲載された記事」で説明する。まずトークンに分割し、

朝日
に
掲載
された
記事

次に各トークンを高次元ベクトル空間に埋め込む embed。

朝日 → [0.2, -0.1, 0.4, ...]
に → [-0.3, 0.2, 0.1, ...]
以下同様...

ちなみに同じ「朝日」でも文脈が異なる、たとえば「朝日が登る」では異なる埋め込みとなる。

なお埋め込みに関し、次元数 (BERT(Base)では 768、GPT-3 では 12288) が上がるほど、より複雑な状況を説明できると予想される。

次に embedding を対象とした自己注意は文脈を学習する。そこで文脈はコロケーション、入力、プロンプト、会話時系列等を契機に構成される。

ここで 2.1 の例に関し、embedding を使い、バグを発見してみる。なお識別しやすいようにすべてを embedding せず、変数のみを対象に embedding 化 (E3='abc', E4='xyz', E5='xyy', E6='Y') する。

```
for E3 in range(len(E6)):
    for E4 in range(len(E6)-1, E3, -1):
        if E6[E4-1] > E6[E4]:
            x = E6[E4-1]; E6[E4-1] = E6[E5]; E6[E4] = x
```

すると E4='xyz'と E5='xyy'の埋め込みの違いからスペルミスのバグが見つかる。

Transformer における「自己注意」が文脈理解に重要な役割を果たすことを最初に提案した論文 を紹介しておく : "Attention Is All You Need" Ashish Vaswani et.al (2017) <https://arxiv.org/abs/1706.03762>

4. LLM と知識の活用

以下に述べる話は、現段階では十分議論されていないが、今後の展開に重要なテーマが含まれている。いずれにしても、今はポイントを記載するに留める。

4.1 AI ロボットへの指示の埋め込み

2.3 に登場するような AI ロボットにおいては、 "pick up"や"put on"等の動作に関する埋め込みに対し、状況把握および操作に関するアフォーダンス等の知識を正解データとして与える必要がある。その設計は難しく、指示者の意図把握、自身の行動予測精度は正解データとして与える知識の量と質に大きく依存する。

4.2 RAG(Retrieval-Augmented Generation)

LLM 自体すでに十分な知識を持っているが、特定の領域の知識は十分ではない。LLM に特定知識を埋め込み Q&A 環境に供するものを RAG と呼び、現在話題になっている。ただし AI ロボットへの指示の埋め込み同様、埋め込みに対応したタスクをシミュレーションする正解データを準備する必要がある (ファインチューニング fine tuning)。それに加え、オントロジーデータのように構造化されたデータを直接埋め込むには、たとえば、Graph Neural Network 等、関係埋め込みを得意とする仕組みを利用するのも一案ではあるが、LLM との協働という面では、互いの出力を自身の入力として利用するという難しさも生じる。

現在、4.1 および 4.2 を課題として研究を進めている。

