ChatGPT

TypeBERT (ESEC/FSE 2021) と ManyTypes4TypeScript (MSR 2022) の背景レビュー

オプショナル型言語TypeScriptの型推論研究として、JesseらはBERTスタイルの事前学習モデル「TypeBERT」を提案し [1]、Mirらは大規模データセット「ManyTypes4TypeScript」を構築・公開している [2]。以下、それぞれの研究におけるモデル規模、学習データ規模、型語彙制限、評価手法などを整理する。

大規模データ vs モデルサイズ

- **TypeBERT**: BERTに類する大規模モデル（エンコーダ層24層・隠れ層次元1024、総パラメータ約3.4億）を用い、JavaScript約2.5万プロジェクトで事前学習、TypeScript約2万プロジェクトで微調整している [3] [4]。これは従来の数百プロジェクト規模のデータセットに比べ格段に大きく、Jesseら自身も「十分なデータがあれば単純なモデルでも最先端を超える性能を示せる」と述べている [1] [5]。実際、TypeBERTは上位100型においてTop1精度約89.5%とLambdaNet（66.9%）を大幅に上回り、全体でも71.12%対64.2%と優位に立った [6] [7]。一方で、この規模のモデルは学習コストも大きく、事前学習には6 GPU×160時間を要したと報告されている [8]。推論時は高速で、バッチ(64×256長)当たり1.28秒（1シーケンス約0.02秒）で処理できるとされる [9]。

- **ManyTypes4TypeScript**: TypeScript向けに新規構築された大規模データセット。13,953プロジェクト・539,571ファイルから約900万件の型注釈を集めており、Python向けデータセットの10倍に相当する規模である [2]。このデータセットは主に学習用に用いられ、HuggingFace上で提供されている。TypeBERTのような巨大モデルだけでなく、CodeBERTやGraphCodeBERT（110–125M程度）など比較的軽量なモデルでも学習・評価できるよう設計されている。

- **軽量モデルの優位／LLMの制約**: TypeBERTの例では高精度が得られたが、モデルサイズ3.4億では依然LLM（例えばGPT-3の1750億）には遠く及ばない。むしろBERT系モデルでは、モデル容量を抑えつつ大規模データで学習するアプローチが注目される。実際、TypeBERTは膨大なデータを消費する一方で、依存関係グラフ不要で高速推論が可能な点も強みである [9]。例えばTypeBERTはGPU一枚で16,384個の型推論を0.02秒/件で行え、LambdaNetが依存グラフ構築に失敗するケースでも問題なく処理できると報告されている [9]。このように、大規模学習を活用しつつも軽量モデルで高精度・高速処理を実現する方向性が示唆されている。

TypeBERTの型語彙制限と評価手法

- **型語彙の制限**: TypeBERTでは出力側に学習データで出現頻度上位40,000種の型のみを語彙として用い、それ以上の型はすべてUNK（未知）ラベルにまとめた [10]。これによりテストデータ中のUNK出現率は約8%に抑えられている（テスト注釈の92%以上をカバー） [10] [11]。ただし語彙を閉じるため、学習データに現れない型（珍しいユーザ定義型など）はUNKとなり、当該位置は誤答とカウントされる。

- **評価指標**: 予測精度の評価にはTop-1/Top-5分類精度を用い、「上位100型」（頻出型）「ユーザ定義型」「その他型」といったカテゴリに分けて測定している [12] [13]。例えばTypeBERTは上位100型でTop1≈89.5%、Top5≈98.5%と非常に高い精度を示し [14]、開発者への推薦候補として実用的な性能

を備えた。一方で、モデルの予測が実際にプログラムで型検査に通るかどうか（型整合性）は評価しておらず、あくまでデータセットの注釈との一致度で性能を判断している。

- **その他の特徴**:　学習・評価時には、情報量が低い「any」型は除外される [10] [15]。TypeBERTはユーザ定義型を含む広い型にも対応し、LambdaNetよりも高い全体Top1精度（71.12% vs 64.2% [6]）を達成した。これは、複雑な静的解析バイアスなしに文脈から型を推測するBERTモデルの表現力の高さを示唆する結果である [16] [7]。

ManyTypes4TypeScriptのデータスキーム

- **データ収集・構造**: GitHub上のTypeScriptプロジェクトをGraphQLクエリで約29,500件収集し、依存パッケージをpnpmで解決してから各ファイルのASTを解析している [17]。AST走査により、人手で注釈された型とコンパイラが推論した型を両方抽出し、コードのトークン列と対応する型ラベルのペアを生成する [17]。データ重複検出ツールで18%の重複コードを排除した後、型注釈付きファイルは約53.96万件に整理された [18]。最終的に13,953プロジェクト、539,571ファイル、9百万件の型注釈からなるコーパスが得られている [2] [19]。

- **データ分割と形式**: コーパスはプロジェクト単位で学習/検証/テストに約80/10/10%の割合で分割される [20]。HuggingFace　Datasets形式で提供され、各サンプルはコードの単語トークン列（JSONの`tokens`リスト）と対応する型ラベル列（`labels`リスト）からなる [21]。この構造により、任意のサブワードトークナイザやTransformerモデルで容易に扱える。

- **型注釈の特徴**:　注釈位置を見ると、変数宣言が約380万件、関数のパラメタが約370万件で最も多い [22]（図2）。型分布では文字列型、`any`型、数値型が上位を占め、特に`any`が非常に多い [23]（図3）。`any`型は有用性が低いため学習・評価から除外されている [24]。推論可能な型割合（型カバレッジ）について、ManyTypes4TSでは「全ての箇所に注釈または推論型があれば静的型付き言語相当になる」と説明しており、現状未注釈部分を確率的手法で埋めることで型カバレッジを飛躍的に高められる可能性がある [25]。

- **型語彙と推論型割合**:　出力語彙は頻度上位50,000種に固定し、それ以上の型はUNKに置換する [24] [26]。この設定で型出現頻度の約94.1%をカバーする（残り約5.9%はUNK扱い） [26]。また、コーパス全体の型のうち約57%がコンパイラ推論型、43%が人手注釈型であり、特に`any`型では人手注釈が20%程度に留まる [27]。これらのデータ構造を用いることで、様々なコード特化モデルが統一的に訓練・評価可能となっている。

以上のように、TypeBERTは**大規模データと高キャパシティモデル**で高い分類精度を達成した例であり、ManyTypes4TypeScriptは**大規模学習用コーパスの標準化**を狙った取り組みである。両者を通じて、先行研究では数万〜数十万規模のコードデータと数億規模のパラメータを組み合わせ、型語彙を上位数万種に限定して評価する手法がとられてきたことが明らかである [3] [26]。これら知見は、BERT系のより軽量なモデル設計においても、**モデルサイズ・データ量・語彙範囲**のトレードオフを考慮する上で重要な背景情報となる。

参考文献: TypeBERT [1] [3] [12] [16]、ManyTypes4TypeScript [2] [17] [26] [28] 他。

---

[1] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] Learning Type Annotation: Is Big Data Enough?
https://www.cs.ucdavis.edu/~devanbu/typebert_esec_fse_.pdf

[2] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] ManyTypes4TypeScript: A Comprehensive TypeScript Dataset for Sequence-Based Type Inference
https://www.kevinrjesse.com/pdfs/ManyTypes4TypeScript.pdf