ChatGPT

TypeWeaverにおける型推論モデルの比較と型検証成功率（ECOOP 2023）

比較対象モデル: DeepTyper・LambdaNet・InCoder

TypeWeaverは任意の型予測モデルを利用できるフレームワークとして設計されており、評価では文献上の3モデル（DeepTyper・LambdaNet・InCoder）が統一パイプラインで比較されています [1]。論文中ではそれぞれ「DeepTyper, a recurrent neural network; LambdaNet, a graph neural network; and InCoder, a general-purpose, multi-language transformer that supports fill-in-the-middle tasks」（DeepTyperは再帰型ニューラルネット、LambdaNetはグラフニューラルネット、InCoderは多目的トランスフォーマーモデル）と位置付けられています [1]。TypeWeaverはこれらのモデルによる型推論結果を取り込み、一貫した手順でTypeScriptコードへの変換・検証を行いました。

型チェック成功率の評価: パッケージ単位 vs ファイル単位

TypeWeaverの評価では、予測された型をコードに挿入してTypeScriptコンパイラで型検査を行い、プロジェクト全体（パッケージ単位）およびファイル単位での**型チェック成功率**が測定されました。結果として、最も性能の良い型予測モデルを用いた場合でも、**パッケージ全体が型検査を通過できたのはわずか21%**に留まりました。一方、**ファイル単位**で見ると**69%**のファイルが型検査を通過するという、粒度による大きな差が報告されています [2]。これは、ファイルごとの型整合性はある程度確保できても、複数ファイルにまたがるパッケージ全体の整合性を保つことがより難しいことを示唆しています。

> “With the best type prediction model, we find that only 21% of packages type check, but more encouragingly, 69% of files type check successfully.” [2]

TypeWeaverの前処理と型注入の必要性

TypeWeaverの著者らは「個々の型を予測するだけでは型移行には不十分」であることを強調しており、予測結果を型検査可能な形にするために必要な**前処理・中間処理**を明示しています [3] [4]。具体的には以下のステップが自動化され、これらを欠いては予測された型でコードを正しく型付けすることはできないと述べられています。

- **依存関係の型定義インポート**: 移行対象のJavaScriptプロジェクトの**依存パッケージ**に型定義が用意されていることを事前に保証する必要があります。 [5] 論文では「Before migrating a JavaScript project, we must ensure that its dependencies are typed. This means transitively migrating dependencies, or ensuring that the dependencies have TypeScript interface declaration (.d.ts) files available.」と説明されており、依存先の型情報（型定義ファイルなど）を導入する処理が最初のステップになります [5]。これはプロジェクトの依存パッケージも含めて型検査を通すための土台となります。

- **モジュール構文の変換**: Node.js向けに書かれたJavaScriptコードがCommonJSモジュール体系を使用している場合、**TypeScript移行後も型情報を正しく維持するため**に、**ECMAScriptモジュール体系（ESM）へ書き換える**必要があります [6]。TypeScriptではCommonJSモジュールを `require` でインポートすると型情報が失われて `any` 型になってしまうためです。そのため、論文でも「to fully

benefit from static type checking, code written with CommonJS modules should be refactored to use ECMAScript modules」（CommonJSで書かれたコードは静的型チェックの恩恵を十分に受けるためにESモジュールに書き換えるべき）と述べられています [6]。

- **型の織り込み（Type Weaving）**： モデルが出力する型ラベルは元のコードに自動で反映されないため、予測結果を**元のJavaScriptソースに統合して型注釈付きのTypeScriptコードを生成**する工程が必要です [4]。著者らはこの工程を「型の**織り込み** (type weaving)」と呼び、「Models that assign type labels to variables do not update the JavaScript source to include type annotations. Therefore, to ask if a program type checks, we must 'weave' the predicted type annotations with the original JavaScript source to produce TypeScript.”（モデルが予測した型ラベルは自動でJSソースに反映されないため、プログラムが型検査を通るか確認するには予測された型注釈を元のJSコードに「織り込んで」TypeScriptに変換する必要がある）と説明しています [4]。このステップにより初めて、予測された型情報がコード上に挿入され、型検証可能な形になります。

- **不正確な予測の除去・フィルタ**: 自然言語処理モデルを用いた型補完では、**生成されたトークン列が型の文法になっていない**ような不適切な出力が混入する場合があります。そのためTypeWeaverでは、**型として無効な予測結果を拒否・クリーンアップする**処理も不可欠です [7]。論文中でも「models... can easily produce token sequences that are not syntactic types. These predictions need to be rejected or cleaned」（モデルは文法的に型でないトークン列を容易に生成し得るため、そうした予測は除去・清掃される必要がある）と述べられています [7]。このフィルタリング工程により、無意味な出力や型エラーの原因となる注釈を事前に排除します。

以上のように、TypeWeaverは単なる型予測に留まらず、予測結果を実際に**型検査可能なコード**にするための一連の工程（依存型の導入・モジュール変換・型挿入・無効出力の除去）を自動化しています。著者らはこれらのステップを踏むことで、初めて自動型付けツールが現実的に機能しうると示しています [8] [9]。

Accuracyから型検証成功率へ：指標としての「type-check可能性」

従来の研究では、予測された型注釈が**正解の型と一致するか**という**Accuracy（正解率）**が指標とされ、高い精度が報告されてきました。しかしTypeWeaverの著者らは、こうした精度指標は実用上**「誤解を招く（accuracy can be misleading）」**可能性があると指摘しています [10]。すなわち、個々の型推論精度が高くてもコード全体が正しく型付けされるとは限らず、重要なのは自動ツールによって生成されたコードが**「TypeScriptの型チェッカを通過できる（passes the TypeScript type checker）」**かどうかであるという立場を明確にしています [10]。TypeWeaverでは型検証に**通ったか否か**を評価軸とすることで、真に有用な型移行の指標を提案しており、論文中でも「predicting individual type annotations is just the first step... We should address a different question: can an automatic type migration tool produce code that type checks?」（個々の型予測はあくまで第一歩に過ぎず、“自動型移行ツールが型検査を通せるコードを生成できるか”という問いに取り組むべきである）と述べられています [3]。これにより、単純な精度よりも**型検査エラーの有無**をもってモデルの有効性を評価するアプローチが提唱されています。

> **出典ガイド**：本回答で引用した内容は、TypeWeaverの研究論文 [11] [2] [5] [6] [8] [7] [3] に基づいています。各引用箇所の末尾に示した【△†Ln-Lm】は論文中の該当ページの行番号を示しており、卒業論文で一次資料として引用する際に参照可能です。例えば、「Accuracyは誤解を生む」との主張はTypeWeaver論文の序論に明記されており [10]、型検証成功率（21%/69%）の数値や前処理ステップの詳細も論文中に明示的に記述されています [2] [8]。以上を踏まえ、TypeWeaver論文の該当部分を一次資料として参照しつつ、本回答を卒論執筆のガイドとしてご活用ください。

---

1 2 10 11 [2302.12163] Do Machine Learning Models Produce TypeScript Types That Type Check?
https://arxiv.org/abs/2302.12163

3 4 5 6 7 8 9 [2302.12163] Do Machine Learning Models Produce TypeScript Types That Type Check?
https://ar5iv.labs.arxiv.org/html/2302.12163