We propose an IDE, with an IDE, model incorporated on-the-fly advice. Due to negligible inference time, the model can introduce specific clauses. Nevertheless, S2S compilers have many directives and clauses. Thus, many automatic source-to-source compilers produce tedious parsing the source code; the limited robustness to the input; and time consuming data-dependence algorithms.

Due to recent innovations in natural language processing (NLP), such as Transformers, the possibility of tackling the task of introducing OpenMP directives with NLP models, rises.

We propose an NLP-based novel parallel code database – that will suggest locations in need for an OpenMP directive and even specific clauses such as private and reduction. We test two source code representations: natural language processing (NLP) and a predefined vocabulary, each token is associated with a numerical vector – the vector in turn is fed to PragFormer.

**Research Objective**

- We propose a model named PragFormer for identifying the need for an OpenMP directive and specific clauses based on the transformer architecture.
- We created a database named Open-OMP with 17,000 unique code snippets containing the for-loop, OpenMP directive (if exists), and an AST representation of the two.
- Half of the code snippets are labeled with an OpenMP directive, while the other half with a high probability, not.
- The code snippets were extracted from GitHub, by searching the phrase “OpenMP”.
- The data-validation-test were split in an 80%-10%-10% ratio.

**OpenMP Directive and Clause Classification**

The following figure presents the result of the classification.

**Future Work**

- Providing the full context of the source code rather than the for-loop segment.
- Exploring new source code representations such as IR2vec.
- Enhancing the model to generate the OpenMP directive.
- Exploring other parallelization paradigms such as MPI and heterogeneous systems.

**PragFormer produces excellent results on OpenMP benchmarks.**

**The following figure presents the result of the explainability tool LIME on two representative examples from the benchmarks:**

- The self-attention mechanism in the transformer architecture calculates a score for each element with respect to any other element. The score determines the amount of consideration between tokens in each consecutive layer.
- Identifying OpenMP directives is mostly hinted from the dependencies between variables and statements. Therefore, the self-attention mechanism is crucial for the task at hand.
- PragFormer is based on the pre-trained model DeepSCC, a fine-tuned RoBERTa model for source code.
- To perform the classification, the transformer architecture feeds its output to an FC layer that predicts a binary label through a softmax layer.

**OpenMP Directive and Clause Classification**

- The prediction error rate as a function of the code length is presented.
- Relatively, examples with length >10 and length <10 produced the same error rate of ~18%.
- It might indicate that length doesn't affect the decision of the model.
- To further test the generality of PragFormer, two benchmarks are tested: SPEC-OMP and PolyBench: