Analyzing Software using Deep Learning

Reasoning about Types and Code Changes with Hierarchical Networks (Part 3)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Overview

- **Hierarchical neural networks**
- **Type prediction**

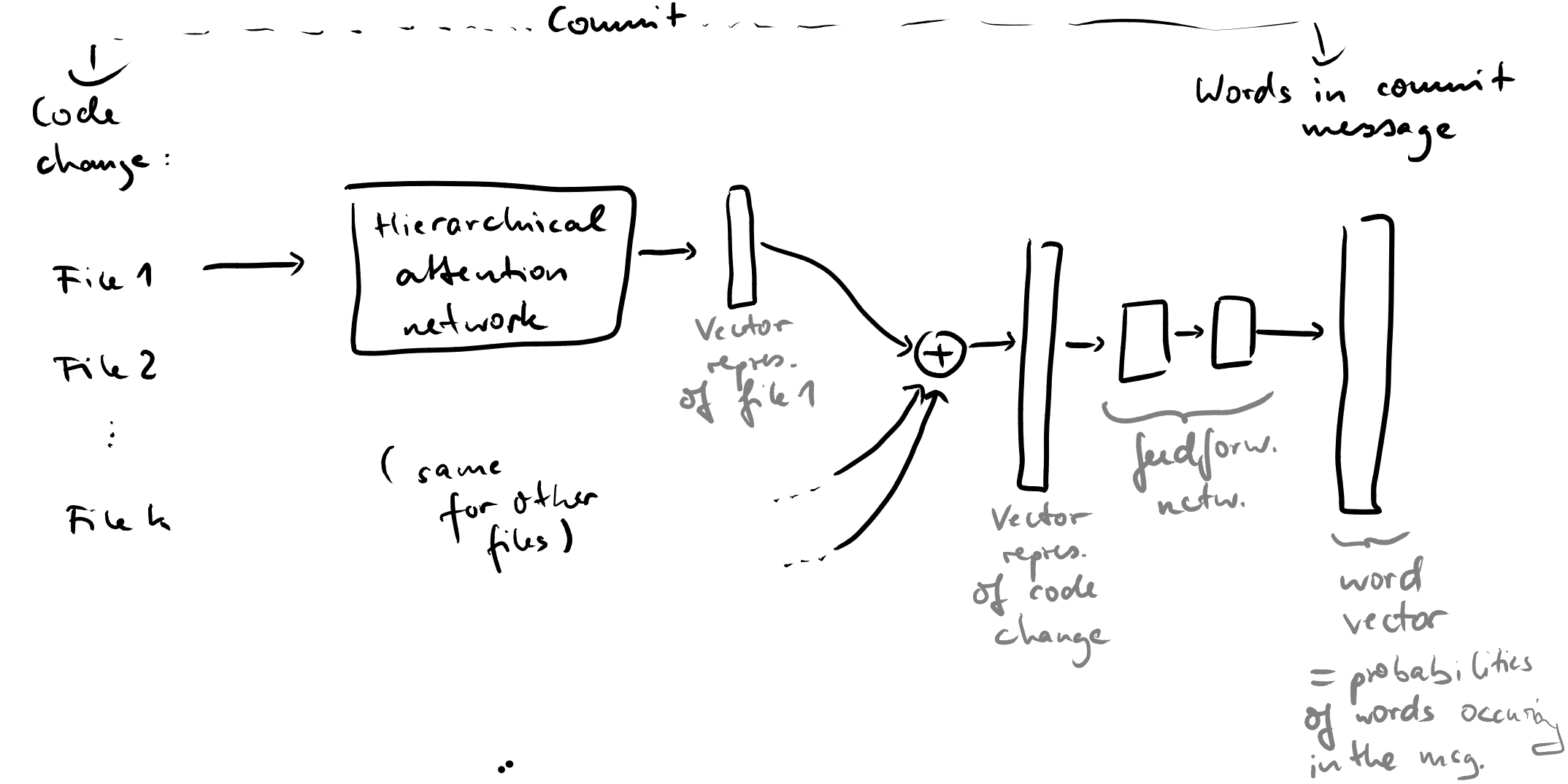
  Based on “TypeWriter: Neural Type Prediction with Search-based Validation” by Pradel et al., 2020
- **Representing code changes** ←

  Based on “CC2Vec: Distributed Representations of Code Changes” by Hoang et al., 2020

Representing Code Changes

- **Source code evolves all the time**
- **Goal: Represent code changes to make predictions**
  - What should be the commit message?
  - Does the change fix a bug?
  - Does the change introduce a bug?

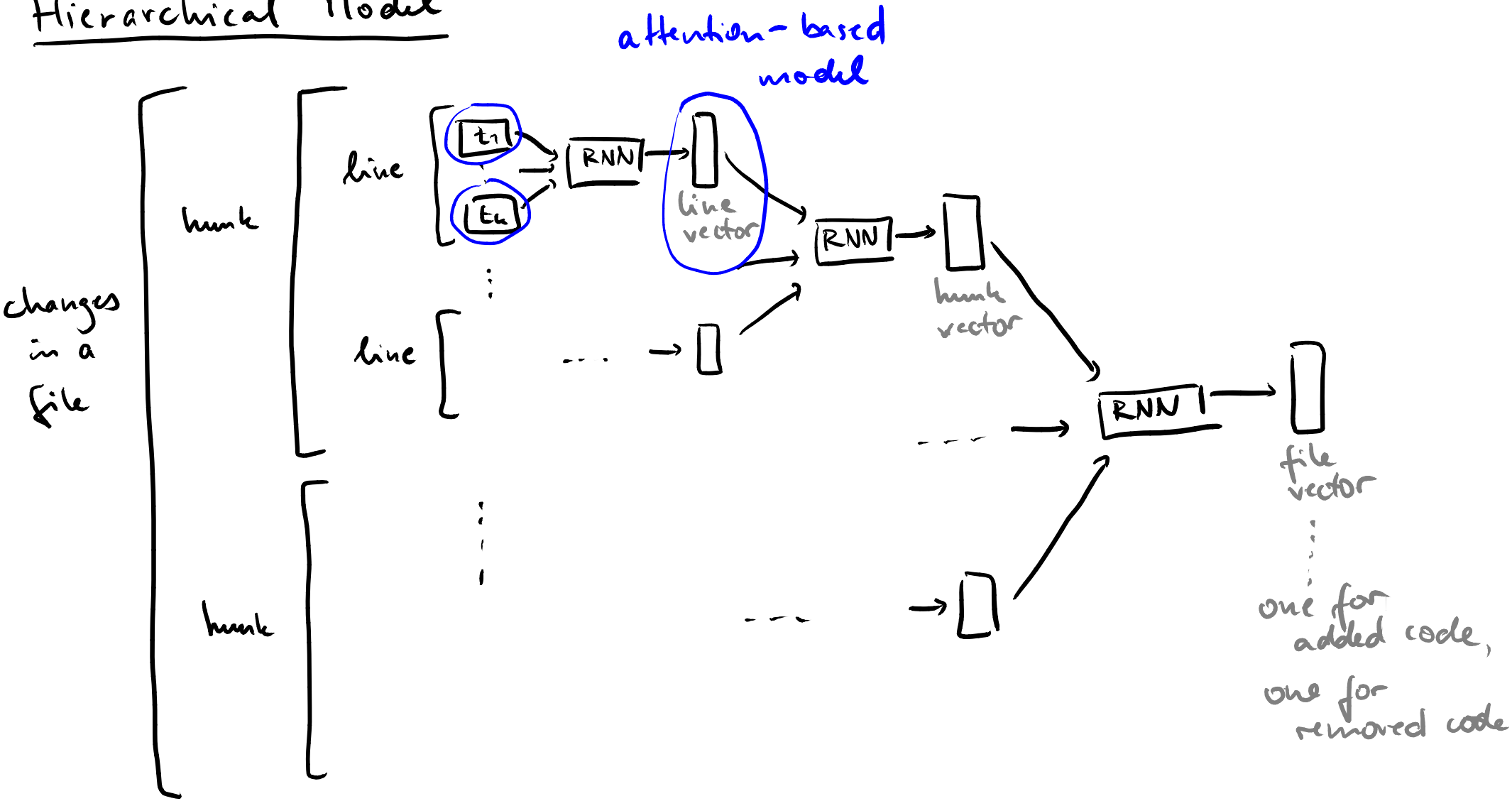
CC2Vec : Overview



Data Extraction

- **Each code change: Set of affected files**
- **Each affected file: Set of hunks**
  - Hunk = consecutive lines of modified code
- **Each hunk: Added and removed lines**
- **Each line: Sequence of code tokens**

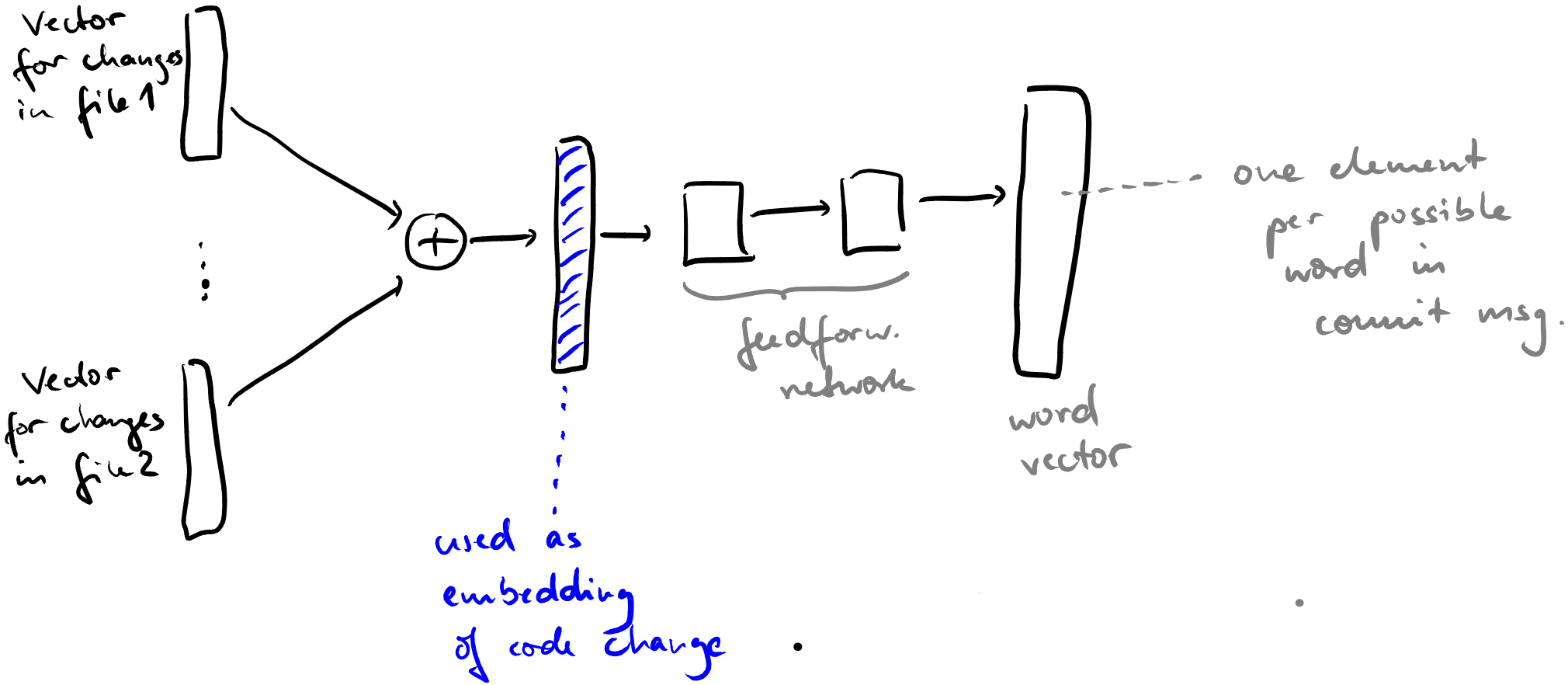
Hierarchical Model



Comparison Layers

- **Goal: Focus on changes in a file**
- **Given: Vector representation of**
  - Added code: e_a
  - Removed code: e_r
- **Set of comparison functions**
  - E.g., element-wise subtraction
- **Result: One vector that summarizes all changes in a file**

Final prediction



Training the Model

- **Gather from version control system of project**
  - Pairs of code change and commit message
  - Evaluation with tens of thousands of pairs
- **Train entire model jointly**
- **Once trained, use embeddings of code changes for specific applications**

Applications

- **Predict commit message**
  - Search for nearest neighbor of code change and reuse it message
- **Predict: Is a code change a bug fix?**
  - Relevant, e.g., to decide which code changes to backport to older Linux kernel versions
- **Just-in-time defect prediction**
  - Useful to allocate quality assurance resources (e.g., code reviews) to code changes