In the paper *code2vec: Learning Distributed Representations of Code*, the author try to solve the task of *semantic labeling of code snippets* by learning a correspondence between the content of a method and a semantic label. The author presents a neural model for representing snippets of code as distributed vectors which can be used to predict semantic properties of the code snippets. The approach of code2vec is: first, extracting syntactic paths based on Abstract Syntax Tree from the code snippet, then represent them as distributed vector representations and use soft-attention mechanism to compute the learned weighted average of the path vectors to produce code vector, finally use the code vector to predict the likely name of the whole snippet. The author evaluate their approach by training a model on a dataset of 14,000,000 methods. The author claims that comariing previous techniques over the same dataset, the code2vec obtains 75% relative improvement and code2vec is the first to successfully predict method names.

The key contributions of this paper are:

1) the neural network that learns code vector (code embeddings).
   The neural network uses a continuous distributed vector representations of code snippets and learns to aggregate multiple syntactic paths into a single vector. The input to the neural network is a code snippet and a corresponding label which expresses the semantic property. The author assume that the distribution of labels can be inferred from syntactic path, therefore, the neural model can learn the label distribution which gives correspondence between code snippet and labels.
2) the benchmark: Quantitative evaluation for predicting cross project method names.
   Code2vec trained on 14,000,000 methods of real-world data. Comparing to previous works which are using LSTMs, CNNs, and CRFs, code2vec achieves 75% relative improvements.

The limitations of code2vec are:

1) code2vec is only able to predict labels that were observed at training time.
   As the targets become specific and diverse, the neural model is unable to compose exact name instead it usually catches only the main idea of the code snippets.
2) code2vec depends on variable names.
   Since the model in trained based on top-starred open source projects which usually have good naming convention. However, when given uninformative and obfuscated variable name, the model is hard to leverage the variable name to predict the target label.

The future work of code2vec:

1) The training of code2vec is joint training means the vector learned through the neural network is tied to the name prediction task. The future work of code2vec is to make the final vector transfer into other applications.
2) Apply the same strategy for the name of classes instead of name of function
3) Beyond the function name generation, utilizing the code vector to understand code snippet is doing