# Project 3 - Semantic Code Search

Professor Pantelis Monogioudis

Zhu, Haoran - hz1922

Li, Qi - ql1045

Zhao, Jingyi - jz2216

## Project Goal

In this project, we use the CodeSearchNet Corpus and participate in the corresponding challenge. We focus only on Python language and the associated dataset contains about 0.5 million pairs of function-documentation pairs and about another 1.1 million functions without an associated documentation. You are required to submit your Normalized Discounted Cumulative Gain (NDCG) score for only the human annotated examples.

We complete this project as following:

1. Read *"CodeSearchNet Challenge Evaluating the State of Semantic Code Search"*
2. Did research on all 5 models implemented on the *CodeSearchNet.*
3. Follow the setup instructions
4. Implement all 5 baseline models

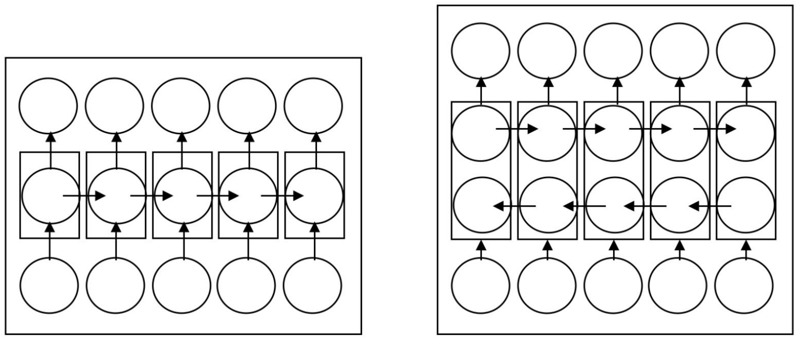
# Algorithms Used Baseline Models

## Neural Bag of Words

## Bidirectional RNN models

Implement the GRU cell from *“On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”* to summarize the input sequence.

**Bidirectional Recurrent Neural Networks (BRNN)** connect two hidden layers of opposite directions to the same output.



(a) (b)

Structure overview

(a) unidirectional RNN
(b) bidirectional RNN

The principle of BRNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states’ output are not connected to inputs of the opposite direction states. The general structure of RNN and BRNN can be depicted in the right diagram. By using two time directions, input information from the past (backwards) and future (forward) of the current time frame can be used unlike standard RNN which requires the delays for including future information.

In CodeSearch, the baseline model default set two rnn layers, LSTM cell, weighted_mean pool mode and run as biRNN. There is a confusion when implementing the baseline model: from the paper, the author used GRUcell, but in the model, the developer used LSTMcell. In the following experiment, we chose to use the developer’s method.

## 1D Convolutional Neural Network

Implement the algorithm from *“Convolutional Neural Networks for Sentence Classification”*.

This is a **CNN with one layer of convolution** on top of word vectors obtained from an unsupervised neural language model. Word vectors trained by word2vec.

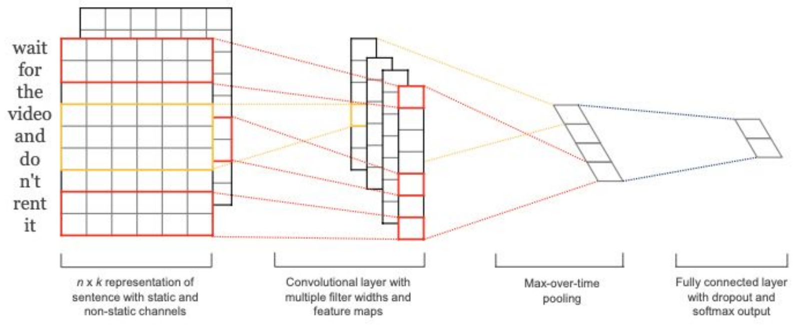

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Model architecture with two channels for an example sentence.

Model detail:

A sentence of length n is represented as $x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n$ , A convolution operation in- volves a filter w is applied to a window of h words to produce a new feature. A feature ci is generated from a window of words xi:i+h−1 by $c_i = f(w \cdot x_{i:i+h-1} + b)$ . Here b ∈ R is a bias term and is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence {x1:h, x2:h+1, . . . , xn−h+1:n} to produce a feature map c = [c1,c2,...,cn−h+1]. Then apply a max-over-time pooling operation over the feature map and take the maximum value of c.

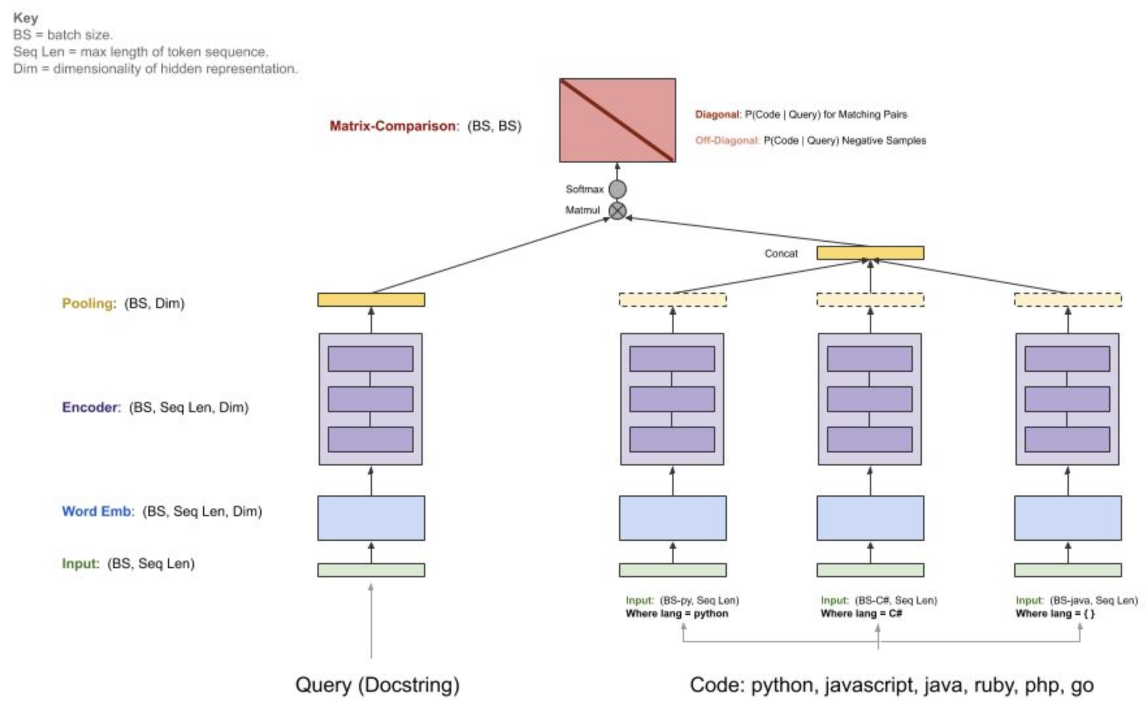
Regularization: dropout on the penultimate layer with a constraint on l2-norms of the weight vectors. instead of using $y = w \cdot z + b$ , for output unit y in forward propagation, dropout uses $y = w \cdot (z \circ r) + b$ , where ◦ is the element-wise multiplication opera- tor and r is a 'masking' vector of Bernoulli random variables with probability p of being 1. Gradients are backpropagated only through the unmasked units.

In CodeSearch, we used tanh as activation function, weighted_mean as pool mode, and softmax as output layer.

## Self-Attention

## 1D-CNN+Self-Attention Hybrid

## Experiment


Key
BS = batch size.
Seq Len = max length of token sequence.
Dim = dimensionality of hidden representation.
Matrix-Comparison: (BS, BS)
Diagonal: P(Code | Query) for Matching Pairs
Off-Diagonal: P(Code | Query) Negative Samples
Softmax
Matmul
Concat
Pooling: (BS, Dim)
Encoder: (BS, Seq Len, Dim)
Word Emb: (BS, Seq Len, Dim)
Input: (BS, Seq Len)
Input: (BS-py, Seq Len)
Where lang = python
Input: (BS-C#, Seq Len)
Where lang = C#
Input: (BS-java, Seq Len)
Where lang = { }
Query (Docstring)
Code: python, javascript, java, ruby, php, go

## Conclusion