Project Part 1 Report

In the class InvertedIndex, two functions were implemented:

index_document(documents),

score(token_list, doc_title).

In the program, several other functions were implemented to help calculate features, train model and predict results from that model.
Spacy, math.log, Numpy, XGBoost and regex were imported to help implement this program.

Features:

1. For a mention in train.pickle, for each candidate of that mention: treat the first 25 (not 'DET' or 'ADV' or 'ADJ') words of this candidate's Wikipedia explanation as a query to the corresponding documents, calculate its tf_idf score in this documents by function InvertedIndex.score().
2. For a mention in train.pickle, for each candidate of that mention: calculate the percentage of document tokens that is missing in the Wikipedia explanation words (by function compute_missing_word_percentage()).
3. For a mention in train.pickle, for each candidate of that mention: calculate the difference between the length of candidate and the length of mentions, and divided by mention length.
4. For a mention in train.pickle, for each candidate of that mention: compute the percentage of characters of candidate that is in mention (in the increasing order, by function compute_mention_candidate_similarity()).

Labels:

Iterate through train_labels.pickle in the order of candidates in train.pickle. Append 1 to Labels to indicate correct result if candidate is the label, otherwise append 0 to indicate incorrect result.

Groups:

Iterate through train.pickle, append length of list 'candidate_entities' in the order of mentions.

XGBoost (or xgb)***:***

Use xgb.DMatrix and xgb.DMatrix.set_group to prepare the train data.

Train model with these parameters: max_depth = 7, eta = 0.05, objective = rank:pairwise, min_child_weight = 0.01, lambda = 100, subsample = 0.5, num_boost_round = 2500 (xgb.train).

Get features from testing file, transform these features into test data, get predictions probabilities from model(xgb.predict). Return the candidate that has the largest probability within a group.

How this program extends part1:

This program uses tf_idf which is similar to part1, but only token tf_idf is considered. The InvertedIndex.score() function considered only first 25 non 'DET','ADV' & 'ADJ' words. Other features didn't appear in part1.