```python
import sys
import math


def train(filename):
    p_dict = {}
    n_dict = {}
    p_count = 0
    n_count = 0
    p_total = 0
    n_total = 0
    with open(filename, "r", encoding="utf-8") as file:
        for line in file:
            line = line.split()
            if line[0] == '+':
                p_count += 1
                for token in range(1, len(line), 2):
                    p_total += int(line[token + 1])
                    try:
                        p_dict[line[token]] += int(line[token + 1])
                    except:
                        p_dict[line[token]] = int(line[token + 1])

            else:
                n_count += 1
                for token in range(1, len(line), 2):
                    n_total += int(line[token + 1])
                    try:
                        n_dict[line[token]] += int(line[token + 1])

                    except:
                        n_dict[line[token]] = int(line[token + 1])

    return p_dict, n_dict, p_total, n_total, p_count, n_count


def vocab(filename):
    V = []
    with open(filename, "r", encoding="utf-8") as file:
        for line in file:
            line = line.split()
            V.append(line[0])

    return V


def parameterize(filename, pDict, nDict, pTotal, nTotal, pCount,
                 nCount, vocab):
    size = len(vocab)

    temp = [token for token in pDict]
    for token in temp:
        if token not in vocab:
            pTotal -= pDict[token]
            del pDict[token]

    temp = [token for token in nDict]
    for token in temp:
        if token not in vocab:
            nTotal -= nDict[token]
            del nDict[token]

    for token in vocab:
        if token not in pDict:
            pDict[token] = 0
        if token not in nDict:
            nDict[token] = 0


    with open(filename, "w", encoding="utf-8") as file:
        #write pos prior then neg prior
        file.write("+ " + str(math.log2(pCount / (pCount + nCount)))
                    + "\n")
        file.write("- " + str(math.log2(nCount / (pCount + nCount)))
                    +"\n")
        # write out log space parameters for an add one smoothed vocab
        for key in pDict:
            file.write("+ " + key + " " +
                        str(math.log2((pDict[key] + 1)/(pTotal + size)))
                        + "\n")

        for key in nDict:
            file.write("- " + key + " " +
                        str(math.log2((nDict[key] + 1)/(nTotal + size)))
                        + "\n")


def test(filename, parameters):
    priors = []
    p_dict = {}
    n_dict = {}
    with open(parameters, "r", encoding="utf-8") as input:
        #pos = 0, neg = 1
        priors = [input.readline().split()[1], input.readline().split()[1]]

        for line in input:
            line = line.split()
            if line[0] == '+':
                p_dict[line[1]] = float(line[2])
            else:
                n_dict[line[1]] = float(line[2])



    with open(filename, "r", encoding="utf-8") as file:
        answers = []
        for line in file:
            p_total = float(priors[0])
            n_total = float(priors[1])
            line = line.lower()
            line = line.split()
            for token in range(1, len(line), 2):
                try:
                    p_total += p_dict[line[token]] * int(line[token + 1])
                    n_total += n_dict[line[token]] * int(line[token + 1])
                except:
                    pass

            if line[0] == '+':
                answers.append(('+', p_total, n_total))
            else:
                answers.append(('-', p_total, n_total))

    return answers


def output(filename, answers):
    correct = 0
    with open(filename, "w", encoding="utf-8") as file:
        for tup in answers:
            file.write("Correct_Label: " + tup[0] + " Pos_Score: "
                        + str(tup[1]) + " Neg_Score: " + str(tup[2]))
            file.write(" Prediction: ")
            if tup[1] > tup[2]:
                file.write("+\n")
                if tup[0] == '+': correct += 1
            else:
                file.write("-\n")
                if tup[0] == '-': correct += 1
        file.write("Accuracy of model: " + str(correct / len(answers)))


if __name__ == "__main__":
    try:
        training_filename = sys.argv[1]
        test_filename = sys.argv[2]
        parameter_filename = sys.argv[3]
        output_filename = sys.argv[4]
        vocab_filename = sys.argv[5]

        parameterize(parameter_filename, *train(training_filename),
                        vocab(vocab_filename))

        output(output_filename, test(test_filename, parameter_filename))

    except:
        print("missing file names in the following order:")
        print("a training file name, a test file name,"
              + " a file name for parameter storage,"
              + " a file name for ouput,"
              + " and a filename for the vocab")
```