Finding Cheating Students

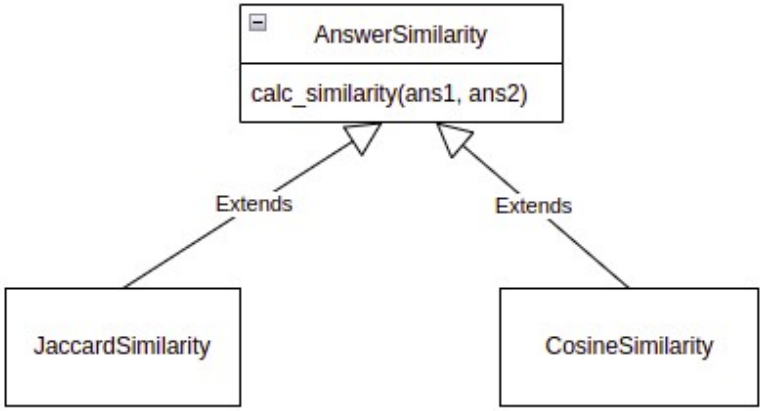
Introduction to information retrieval

Fatemeh Karimi

Project structure

- Detecting answer similarities
- Calculating person similarities
- Detecting cheats
- Displaying results

Detecting answer similarities

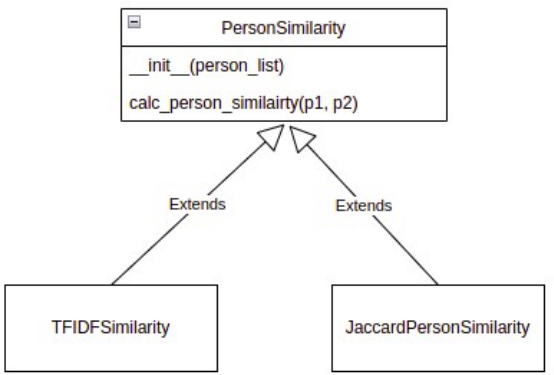



```python
class JaccardSimilaritiy(AnswerSimilarity):
    def calc_similarity(self, ans1, ans2):
        count_common = 0
        for w in ans1.keys():
            if w in ans2.keys():
                count_common += 1
        count_union = len(ans1.keys()) + len(ans2.keys()) - count_common

        if count_union == 0:
            return 0
        return count_common / count_union
```

```python
class CosineSimilarity(AnswerSimilarity):
    def calc_similarity(self, ans1, ans2):
        if len(ans1) != len(ans2):
            raise Exception("vectors should have equal size")
        dot_product = np.dot(ans1, ans2)
        norm_vec1 = np.linalg.norm(ans1)
        norm_vec2 = np.linalg.norm(ans2)

        if norm_vec1 == 0 or norm_vec2 == 0:
            return 0
        return dot_product / (norm_vec1 * norm_vec2)
```

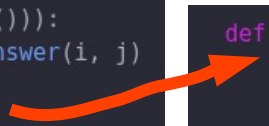
Calculating person similarity

```python
def build_tf_matrix(self):
    result = []
    for person in self.person_list:
        person_tf = []
        for ans in person.get_ans_list():
            tf_matrix = dict(
                zip(self.word_list, [0] * len(self.word_list)))
            word_freq = ans.get_word_freq()
            for word in word_freq.keys():
                tf_matrix[word] = word_freq[word]['count']
            person_tf.append(tf_matrix)
        result.append(person_tf)
    return result

def build_idf_matrix(self):
    df_matrix = dict(zip(self.word_list, [0] * len(self.word_list)))
    total_ans = 0
    for person in self.person_list:
        for ans in person.get_ans_list():
            for word in ans.get_word_freq().keys():
                df_matrix[word] += 1
            total_ans += 1

    result = dict()
    for word, ans_freq in df_matrix.items():
        result[word] = np.log(total_ans / ans_freq)
        if result[word] == 0:
            result[word] = np.log(total_ans / (total_ans - 1)) / 2
    return result
```



Detecting cheats

```python
def find_cheated(self):
    result = {}
    for i in range(len(self.person_list)):
        person_cheat = {}
        person_cheat_with_set = set()
        for j in range(len(self.person_list[i].get_ans_list())):
            sim_person, sim_value = self.calc_max_sim_for_answer(i, j)
            if sim_value >= Settings.MIN_CHEAT_BOUNDRY:
                person_cheat[str(j)] = {
                    'similarity': sim_value,
                    'with who': sim_person}
                person_cheat_with_set.add(sim_person)
        suspects = []
        for who in person_cheat_with_set:
            suspects.append({
                'id': who,
                'name': self.person_list[who].name
            })
        report = {
            'suspects': suspects,
            'details': person_cheat
        }
        result[str(i)] = report
    return result
```



```python
def calc_max_sim_for_answer(self, person_idx, ans_idx):
    max_sim_value = 0
    sim_person_idx = person_idx
    for i in range(len(self.person_list)):
        if i == person_idx:
            continue
        sim_value = self.sim_matrix[person_idx][i][ans_idx]
        if sim_value > max_sim_value:
            max_sim_value = sim_value
            sim_person_idx = i
    return (sim_person_idx, max_sim_value)
```

Displaying results