Author Disambiguation Algorithm

Input: Set of PDF papers
Output: Set of paper clusters where each cluster corresponds to a distinct author. Each paper in a cluster should come with a confidence score, which should be based on how the prediction was made

Step 1. For each paper, extract the title, the list of author names, and the list of affiliations using a multimodal language model (e.g., gpt-4o) by giving the model a screenshot of the first page of the paper as well as a textual prompt describing the task. The task of outputting the affiliation of each author is too difficult for the gpt-4o, but outputting a list of affiliations or a list of authors is easy.

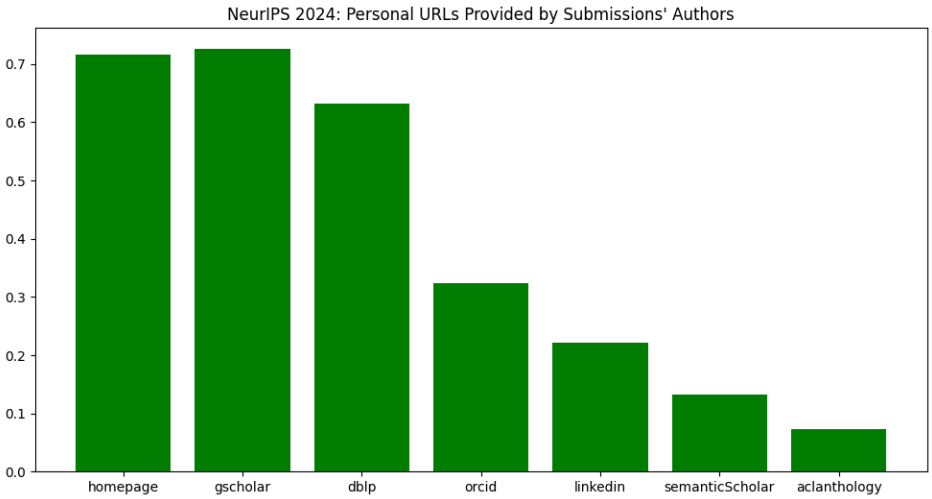
Step 2. Determine if each paper is on OpenReview or not by submitting a query to a search engine API. The URL should have the following format:
https://openreview.net/forum?id={paper_id}
If the paper is on OpenReview, go to Step 3a. Otherwise, go to Step 3b.

Step 3a. Make a GET request to get the content of the OpenReview paper page and extract the author IDs. Using the author IDs, we can construct the URLs of the OpenReview profiles for each author, which contain information such as the names that each author published under, career and education history, and personal links provided by each author. Note that even if the correct OpenReview paper page is found, this process is still fallible (e.g., the author ID can occasionally be a DBLP link instead of the actual username or a OpenReview profile may be inactive).

- Comments
    - Each OpenReview profile can serve as a unique identifier for an author
    - We can get the affiliation of a user if we have their author ID (and if their profile is up-to-date). Note that the affiliation may not match the affiliation from the paper, but I think the goal should be to get the current affiliation instead of the affiliation mentioned in the paper since it is more helpful for identifying the researcher.
    - Currently, users of OpenReview can optionally provide the following personal links: personal website/homepage, Google Scholar profile, DBLP, ORCID, LinkedIn, Semantic Scholar, ACL Anthology

Step 3b. If a paper is not hosted on OpenReview, we will need to find a reliable URL that can link together an individual and the paper (e.g., a personal website that has the paper listed under a publications section). To determine which sources to search

for, 2,405 OpenReview users who made submissions to NeurIPS 2024 were sampled and used to generate the following figure, which shows the proportion of users that provided each type of personal link on their OpenReview profiles.
From the chart, we can see that most authors chose to share a homepage and a Google Scholar page, and interestingly 88.94% of the sample have at least one of these two types of URLs on their profiles. This shows that these two types of URLs are heavily used by researchers in the AI community, and we can therefore consider these sources reliable. Also, note that this analysis does not say anything about the behavior of computer science researchers outside of AI, but is sufficient since we're evaluating on AI papers.

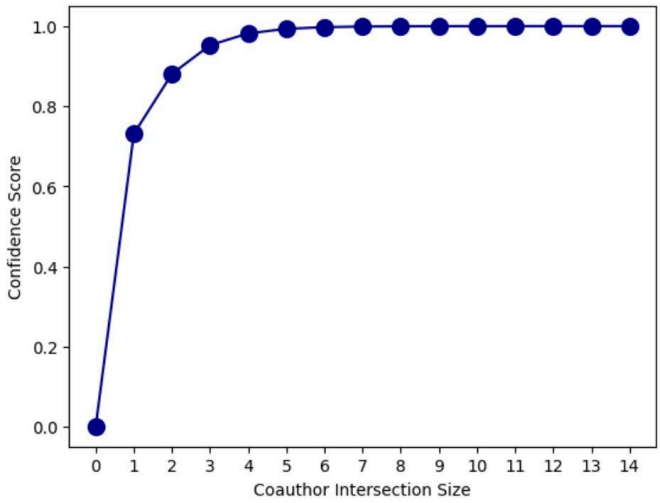


Searching for personal website. Recall that in Step 1, we extracted the title, the list of authors, and the list of affiliations from the PDF. We don't know which authors are affiliated with which institutions, but we can use these lists to help us formulate search queries for finding the authors' personal websites. Also note that each search result from the API contains a title, a link, and a snippet.
For each author name, we can search for "{author name} {institution name}" by plugging in every institution name in the list of affiliations. A search result can be considered a match if (1) the author's name appears in the title and if (2) the link/URL ends with ".github.io" or contains ".edu". If it is not possible to find a search result that satisfies both of these criteria, we can try to first find results that satisfy the first criterion and then feed the gpt-4o a screenshot of the page at a URL to have it determine if the page is indeed a personal website of a computer science researcher with the author name that we're considering.
Once we find the URL of the personal website, we can submit another query to the search API by providing "{website URL} {paper title}" to find a search result where (1) the result's link is the website URL from the query and (2) the paper title is a substring of the result's snippet. Finding this search result proves that the paper is

mentioned somewhere on the personal website, which allows us to conclude that the owner of the website is the author of the paper.

Searching for google scholar profile. To find the google scholar profile of an author, we can submit the following query to the search API: "{author name} google scholar {paper title}". The search result that we're looking for should have the author name in the result's title, "scholar.google" in the link, and the paper's title somewhere in the snippet.
At the end of this step, author names from papers can be matched with either a google scholar profile, a personal website, both, or neither.

Step 4. Attempt to match authors from papers that aren't hosted on OpenReview with authors from papers that are hosted on OpenReview by checking for equivalence between Google Scholar URLs or personal websites. For authors that don't have any corresponding URLs, we can see if their names match names of "identified" authors and then determine if they are the same person by looking at the similarities between their papers' co-authors (decisions made in this way will lead to lower confidence scores). In more detail, suppose we are trying to determine if Author A and Author B are the same person in the real world, where Author A has corresponding URLs and Author B does not. We can take the set of coauthors of Author A and the set of coauthors of Author B, look at the intersection between them, and find the cardinality of this set. If the size of the intersection is zero, then we have no reason to believe that they are the same person. However, if the size is at least one, our confidence score (between 0 and 1) should be nonzero and increase monotonically with the size, saturating quickly at the beginning. The sigmoid function captures this perfectly, so this is employed as the confidence score for inputs that are greater than 0 (see figure below).



Therefore, if Author A and Author B have at least one coauthor in common, we merge the profiles and say that they are the same person with some confidence score that is a function of the number of coauthors that they have in common. Since the output of the system is a set of paper assignments, the confidence score from merging profiles is translated into confidence scores for the assignments. All papers that were originally assigned to Author A will receive a score of 1, while all papers that were originally assigned to Author B will receive the confidence score from merging the profiles.

Implementation

In order to run the code, you will need the following:

1. An OpenReview account
2. A serper.dev API key
3. An OpenAI API key
4. An INI file that contains the following content (just create a file with ".ini" as the extension):

```
[BASIC]
USERNAME = <OpenReview Username>
PASSWORD = <OpenReview Password>
SERPER_API_KEY = <Serper API Key>
OPENAI_API_KEY = <OpenAI API Key>
```

The algorithm takes as input a path to a directory that contains the PDF papers. Each paper should have a unique ID, and should be named as follows: <unique_ID>.pdf. To run the algorithm, run "pip install -r requirements.txt" in a new virtual environment and then run two CLI commands:

```
python3 cli.py --part 1 --pdf_dir <path_to_pdf_dir> --tmp_dir <tmp_dir> --save_dir <save_dir> --credentials_path <path_to_ini>

python3 cli.py --part 2 --pdf_dir <path_to_pdf_dir> --tmp_dir <tmp_dir> --save_dir <save_dir> --credentials_path <path_to_ini>
```

Note that everything except for "--part" is the same. "--tmp_dir" should be the path to a directory that will be created by the program, and will be written to during execution. "--save_dir" is the path to the directory (created by the program) that will contain the output of the system (two CSV files). "--credentials_path" is the path to the INI file that was mentioned earlier.

The first CLI command will extract the title, list of authors, and list of affiliations from each PDF paper and store the information in a CSV file. The second CLI

command will run the rest of the algorithm and create two CSV files in the "--save_dir" directory: authors.csv and assignments.csv.
Each row in the table in authors.csv will correspond to an author identified by the system and store their URLs (three possible types) and affiliation if this information was found. Every author has a unique ID.
The table in assignments.csv maps author IDs from authors.csv to PDFs and provides a confidence score for each assignment.

Demo

To provide examples of how the system works, there are a few input and output directories in the repository.

Example 1. "testdata_mini1" contains four PDFs from ICLR 2024, which collectively contain 21 unique authors. Two of the authors (Kun Zhang and Jiang Bian) are authors of two papers while all other authors are authors of just one paper. From authors.csv in "testdata_mini1_out", we can see that the system found the correct OpenReview profiles and paper assignments for all authors. Additionally, all paper assignments have confidence scores of 1 because we were able to find OpenReview profiles for all authors.

Example 2. "testdata_mini2" contains two papers that aren't hosted on OpenReview. There are four unique authors in the set, and two of them (Reza Shokri and Yair Zick) are authors of two papers, while the other authors are authors of just one. From authors.csv, we can see that there is one row for each author and since the papers aren't on OpenReview, there are no OpenReview URLs for any of the authors. However, the system was able to find the google scholar profiles and affiliations of all four researchers. The homepage URL of Yair Zick is also correct. From assignments.csv, we have correct assignments for both papers, but one assignment received a confidence score of 0.73, which happened because two profiles were appropriately merged.

Example 3. "testdata_mini3" contains four papers from ICLR 2024, which collectively contain 31 unique authors (however, only 29 authors are listed as corresponding users of the papers on OpenReview). One author (Luke Zettlemoyer) is on two papers while all other authors are on just one. According to authors.csv, the system was able to find 29 authors, but 6 of them don't have any corresponding URLs. However, all of the identified URLs (OpenReview, Google Scholar, personal website), affiliations, and paper assignments are correct.