Norrec Nieh & Jason Zhang
CS5800 Fall 2022, Project Final Report

SUMMARY

We chose to implement our programs in Python, based on our knowledge of the pandas library (which we describe in the Technical Discussion section), as well as our familiarity with the language. We attempted to match the Avalara and UNSPSC tax codes in a variety of ways, which we have distilled in the following report into four main iterations representing the methodologies we developed chronologically throughout the project. In the first three iterations, we adjusted the size of the inputs in order to discern acceptable tradeoffs between accuracy and efficiency. Our final code completed the matchmaking process in 59.3 minutes, and generated 2535 matches (out of 2544) between the Avalara and UNSPSC categories, 30% of which are correct matches.

TECHNICAL DISCUSSION

We used the pandas library[1] to read in the Avalara and UNSPSC data, representing each file as separate data frames[2] such that we could parse each frame to create our data structures. Among the functions defined by pandas, 'read_excel'[3] was used to read our .xlsx files into the data frames, and iterators were used to extract pertinent data from cells. We also used the 'time' function from the time library[4] in order to record the seconds taken to accomplish each subtask. *Figure 0* below shows the related outputs generated by the 'time' function that we used to analyze the performance of each iteration. Additionally, in all iterations we used the 'findall' function in the re (regex) library[5] to split our description strings by any non-alphabetic characters, and the 'isnan' function in the math library to check whether a pandas data frame cell held a 'nan' value.[6]

Below, we will detail in brief the methodologies behind each of our four iterations.

```
>> Converting UNSPSC file to data frame...
>>>> Conversion complete. Process took 19.91 seconds.

>> Building UNSPSC tree...
>>>> Build complete. Process took 7.2 seconds.

>> Converting Avalara file to data frame...
>>>> Conversion complete. Process took 0.44 seconds.

>> Building Avalara hash map...
>>>> Build complete. Process took 0.04 seconds.

>> Beginning matching...
>>>> Matching complete. Process took 36.33 seconds.

>>>> 705 matches at the commodity level made.

>>>> 1350 matches at other levels made.

>>>> 455 matches failed at the highest level.
```

figure 0: Status output after running iteration 1.

---

[1] "Pandas Documentation." Pandas. Version 1.5.2. Nov 22, 2022. https://pandas.pydata.org/docs/
[2] "pandas.DataFrame." Pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
[3] "pandas.read_excel." Pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html
[4] "time – Time access and conversions." Python. Version 3.11.1. https://docs.python.org/3/library/time.html
[5] "re – Regular expression operations." Python. Version 3.11.1. https://docs.python.org/3/library/re.html
[6] "math – Mathematical functions." Python. Version 3.11.1. https://docs.python.org/3/library/math.html

Iteration 1: Tree Traversal

In our first iteration (see the /iter1 folder), we built a tree data structure to hold the UNSPSC data, where segment, family, class, and commodity nodes formed a distinct hierarchy, with each commodity being expressed as a leaf node (see UNSPSC_structure_tree.py). Each node at any level has as attributes an ID for the segment, family, class, or commodity, and a list containing all words greater than length 3 found in the commodity title and description, represented in lower case. We used a Python dictionary to hold the Avalara data, where the key was the tax code and the value was the list of words corresponding to the tax code description (see Avalara_structure_tree.py).

For each entry in the Avalara dictionary, we begin at the highest level of the UNSPSC tree, matching each word in the Avalara item's word list to each word in each UNSPSC node's word list at that particular level. The node with the highest number of matches will be chosen, and we will similarly compare all of that node's children. This will repeat until a leaf node is found (see matchmaker_tree.py, represented in *figure 1a* as pseudocode).

```
ITERATION 1

def matchAll():

    for item in Avalara_dict:
        iterator = UNSPSC_tree

        while iterator is not a leaf node:
            for child in iterator.child_list:

                for word in item.word_list:
                    for comparator in child.word_list:
                        if word == comparator:
                            curr_count += 1

                if curr_count > max_count:
                    max_count = curr_count
                    max_child = child

            iterator = max_child
```

figure 1a: Pseudocode for iteration 1

Using this approach, we made 705 matches out of 2544, with an accuracy of about 17% (see results_tree_1.txt). The matching process took 36.33 seconds on a 2.3 GHz 8-Core Intel Core i9 processor (MacBook Pro 2019). Below in *figure 1b* we can see a sample of the results. While some correct matches did occur, this depended on a perfect match at the top level, which was an uncommon occurrence. Note that the first two items in the screenshot were not matched.

```
PF050416    Popcorn or candy in a basket or tin wherein exempt item items constitute 90% to 100% of the total value of the container
PF050417    Popcorn or candy in a basket or tin wherein exempt item items constitute 51% to 89% of the total value of the container
PF050418    Food combo pack #16 - food 51%-74%, dietary supplements 26%-49% 178682  50501703    ['nutritional', 'protein', 'supplement']
PF050419    Food combo pack #17 - food 75%-89%, dietary supplements 11%-25% 178682  50501703    ['nutritional', 'protein', 'supplement']
PF050420    Food combo pack #18 - food 90%-99%, dietary supplements 1%-10%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050421    Food combo pack #19 - food 1%-50%, dietary supplements 50%-99%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050422    Food combo pack #20 - food 0%-50%, dietary supplements 1%-89%, hard goods 11%-49%   178682  50501703    ['nutritional', 'protein', 'supplement']
PF050423    Food combo pack #21 - food 0%-50%, dietary supplements 1%-99%, hard goods 1%-10%    178682  50501703    ['nutritional', 'protein', 'supplement']
PF050424    Food combo pack #22 - food 0%-50%, dietary supplements 0%-50%, hard goods 50%-100%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050425    Food combo pack #23 - food 51%-74%, dietary supplements 26%-49%, hard goods 1%-10%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050426    Food combo pack #24 - food 75%-89%, dietary supplements 1%-24%, hard goods 1%-10%   178682  50501703    ['nutritional', 'protein', 'supplement']
PF050427    Food combo pack #25 - food 51%-74%, dietary supplements 1%-24%, hard goods 25%-48%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050428    Food combo pack #26 - food 75%-88%, dietary supplements 1%-14%, hard goods 11%-24%  178682  50501703    ['nutritional', 'protein', 'supplement']
PF050429    Food combo pack #27 - food 90%-98%, dietary supplements 1%-9%, hard goods 1%-9% 178682  50501703    ['nutritional', 'protein', 'supplement']
PF051498    Candy with an item of nominal value - candy must be the predominant value
```

figure 1b: Sample results for iteration 1, which were printed to results_tree_1.txt

Seeing the small number of matches, we made an adjustment so that a failed match where no matches were found in any child's word lists would return the parent node's ID as a result. Though this returned 1350 more matches, these extra matches were not at the commodity level and so ultimately did not improve the quality of the results (see results_tree_2.txt).

Iteration 2: <u>Rabin-Karp</u>

In our second iteration (see the /iter2 folder), we went with a brute force approach, an approach contingent on us finding a more efficient algorithm to match descriptions. We decided to implement the Rabin-Karp string-searching algorithm, which determines a hash value for a pattern and locates that pattern in longer strings.[7] We have attempted to depict the pseudocode in *figure 2a:*

```
def rabinKarp(pattern, text):

    d = 26 # num of letters
    q = 5381 # large prime number
    m = length of pattern
    n = length of text
    h = 1
    num_matches = 0

    for i in range (0 -> m - 1):
        h = (h * d) % q

    for i in range (0 -> (n - m + 1))
        if p == t:
            for j in range (0 -> m):
                if text[i + j] != pattern[j]:
                    break
            j += 1
            if j == m:
                num_matches += 1

        if i < n - m:
            t = (d * h * (t - ord(text[i]) + ord(text[i + m])) % q

            if t < 0:
                t = t + q

    return num_matches
```

figure 2a: Pseudocode for Rabin-Karp algorithm

Using this approach, we ran the matchmaker program on two versions of inputs. In the first version, we built the Avalara data structure such that when iterating through the Avalara data frame, we distinguish between a "parent" category and a "child" node (see Avalara_structure.py). We took a conservative approach to add the additional information of the parent category to the Avalara item's word list if the title of the item includes all the words of the parent item's title. This was to increase the amount of data we had to compare to the UNSPSC descriptions. Meanwhile, for each UNSPSC item, we used all the words found in the segment, family, class, and commodity titles and descriptions. In *figure 2b* we see the line-by-line matching as opposed to iteration 1's tree traversal, while the actual matching is outsourced to the rabinKarp function:

[7] "Rabin-Karp Algorithm for Pattern Searching." Geeksforgeeks. Sep 23, 2022. https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/

```
ITERATION 2

def rabinKarpMatch():

    for ava_item in Avalara_dict:
        for unspsc_item in UNSPSC_dict:
            for word in ava_item.word_list:
                curr_count = rabinKarp(word, unspsc_item.string)

            if curr_count > max_count:
                max_count = curr_count
                max_item = unspsc_item
```

figure 2a: Pseudocode for iteration 2

Despite the efficiency of the Rabin-Karp algorithm, in this version the matching process ran for days on a Microsoft Azure V2 virtual machine running Windows 11 with 2 cores and 8 GB of memory. We estimate completion at 13.8 days (up to 8 minutes for each item), because we are basically looking through every string in the UNSPSC dictionary for each word in the Avalara dictionary. Based on a sample size of 161 items, we assume 2277 matches made out of 2544, with an accuracy of about 24% (results_rk_1.txt). We can see in *figure 2b* that some Avalara items are matched well, but at the same time similar items in computer software are matched to healthcare services. We think that because we used extremely large inputs, we had a lot of bad matches because meaningless words have the same priority as meaningful words.

DC010000	Computer software (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC010100	Custom computer software - physical media (business-to-business)	is	matching	with	85301801		healthcare	services	this	seg
DC010200	Custom computer software - electronically downloaded (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC010300	Custom computer software - load and leave (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC010400	Computer software (prewritten/canned) physical media (business-to-business)	is	matching	with	85301801		healthcare	services	this	seg
DC010500	Computer software (prewritten/canned) electronically downloaded (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC010600	Computer software (prewritten/canned) delivered via load and leave (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC011000	Computer software system software	is	matching	with	85282202		healthcare	services	this	seg
DC020000	Computer software (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020100	Custom computer software - physical media (business-to-customer)	is	matching	with	85301801		healthcare	services	this	seg
DC020200	Custom computer software - electronically downloaded (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020300	Custom computer software - load and leave (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020600	Computer software - (prewritten/canned) delivered via load and leave (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020400	Computer software - (prewritten/canned) physical media (business-to-customer)	is	matching	with	85301801		healthcare	services	this	seg
DC020402	Computer software - non-educational - prewritten/canned - physical media (business-to-business)	is	matching	with	85301801		healthcare	services	this	seg
DC020500	Computer software - (prewritten/canned) electronically downloaded (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020501	Computer software - educational - prewritten/canned - electronically downloaded (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020502	Computer software - non-educational - prewritten/canned - electronically downloaded (business-to-customer)	is	matching	with	43211506		information	technology	broadcasting	anc
DC020503	Computer software - non-educational - prewritten/canned - electronically downloaded - (business-to-business)	is	matching	with	43211506		information	technology	broadcasting	anc

figure 2b: Sample results for iteration 2, version 1 (titles and descriptions)

Responding to the poor efficiency of the first version, we ran a second version on an identical virtual machine, matching only the titles (and not the descriptions) of the Avalara and UNSPSC items (see Avalara_structure_titlesonly.py; the appropriate lines are simply commented out in UNSPSC_structure_dict_unsorted.py). In this version, we ran the program for 4 hours, and estimated completion at 33 hours. Based on our partial result of size 302, we expect 2344 matches out of 2544, with an accuracy of about 5% (results_rk_2.txt). We can see in *figure 2c* that digital goods have been matched to computer software licensing rental or leasing services. While some of our matches are in the vicinity of the right categories, there is in many a case too little information given to make an exact match.

26	DC010000is	Digital goods: photographs	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
27	DC010100is	Digital goods: photographs-physical media	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
28	DC010200is	Digital goods: photographs-streaming / electronic download	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
29	DC010300is	Photographs - streaming / electronic download *see additional code description	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
30	DC010400is	Digital goods: videos	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
31	DC010500is	Digital goods: videos-physical media	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
32	DC010600is	Digital goods: videos-streaming / electronic download	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
33	DC011000is	Video programming streamed over the internet	matching	with	43232614	Information Technology Broadcasting and Telecommunications	Computer aided design CAD and computer aided manufacturing CAM system
34	DC020000is	Videos - streaming / electronic download *see additional code description	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
35	DC020100is	Digital audio visual works	matching	with	43232303	Information Technology Broadcasting and Telecommunications	Customer relationship management CRM software
36	DC020200is	Digital audio visual works sold to an end user with rights for permanent use	matching	with	43232303	Information Technology Broadcasting and Telecommunications	Customer relationship management CRM software
37	DC020300is	Digital audio visual works (with rights of less than permanent use)	matching	with	43232303	Information Technology Broadcasting and Telecommunications	Customer relationship management CRM software
38	DC020600is	Digital audio visual works (with rights conditioned on continued payments)	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
39	DC020400is	Digital audio visual works (with rights of less than permanent use) *see additional avatax system tax code information	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
40	DC020402is	Digital audio visual works sold to users other than the end user	matching	with	43232505	Information Technology Broadcasting and Telecommunications	Customer relationship management CRM software
41	DC020500is	Freight	matching	with	81112501	Engineering and Research and Technology Based Services	Computer software licensing rental or leasing service
42	DC020501is	Delivery by company vehicle	matching	with	43232502	Information Technology Broadcasting and Telecommunications	Computer based training software
43	DC020502is	Delivery by company vehicle before passage of title	matching	with	43232502	Information Technology Broadcasting and Telecommunications	Computer based training software
44	DC020503is	Delivery by company vehicle after passage of title	matching	with	43232502	Information Technology Broadcasting and Telecommunications	Computer based training software
45	DC060000is	Shipping only (not paid directly to common carrier)	matching	with	43231512	License management software	License management software
46	DC070000is	Shipping only common carrier - fob destination (backward compatibility)	matching	with	78131806	Transportation and Storage and Mail Services	Self storage or mini storage service
47	DD020000is	Shipping only common carrier - fob destination	matching	with	42211509	Medical Equipment and Accessories and Supplies	Head or face protective helmet or device for the physically challenged
48	DD040000is	Shipping only common carrier - fob origin	matching	with	42211509	Medical Equipment and Accessories and Supplies	Head or face protective helmet or device for the physically challenged
49	DG010000is	Shipping only non-common carrier - fob destination	matching	with	60141101	Musical Instruments and Games and Toys and Arts and Crafts and	Educational games

figure 2c: Sample results for iteration 2, version 2 (titles only)

<u>*Iteration 3:* Sorted Inputs (Prototype)</u>

For our third iteration (see the /iter3 folder), we abandoned the Rabin-Karp algorithm in favor of sorting the word lists on both the Avalara and UNSPSC sides. For each word in an Avalara item word list, we would iterate through the UNSPSC item word list until we find a match. As we can see in *figure 3a*, if there is a match, then we would search for the next word in the Avalara list through the rest of the UNSPSC list. Otherwise, if no matches are found, we simply iterate to the end of the UNSPSC list in an effort to find a first match. While this approach would be refined in our final iteration, in this iteration we were primarily concerned with outputting more accurate matches at a faster speed. The idea is to have quantitatively less but qualitatively better matches.

```
ITERATION 3

def sortedMatch():

    for ava_item in Avalara_dict:

        max_count = 0, max_item = None

        for unspsc_item in UNSPSC_dict():

            while ava_index <= ava_lastindex and unspsc_index <= unspsc_lastindex

                while ava_index <= ava_lastindex:
                    while unspsc_index <= unspsc_lastindex:

                        if ava_item.wordlist[ava_index] == unspsc_item[unspsc_index]:
                            curr_count += 1
                            break
                        else:
                            unspsc_index += 1

                    ava_index += 1

            if curr_count > max_count:
                max_count = curr_count
                max_item = unspsc_item
```

figure 3a: Pseudocode for iteration 3

Whereas previous iterations took around 7 seconds to build the UNSPSC dictionary, sorting the word lists increased the time taken to accomplish this subtask to around 20 seconds. However, it vastly improved the efficiency of running the program. When using full descriptions on the UNSPSC side and only the titles on the Avalara side, this approach yielded 1892 matches out of 2544, with an accuracy of about 25% (results_sorted_1.txt). The matching process took 40.2 minutes on a 2.3 GHz 8-Core Intel Core i9 processor (MacBook Pro 2019), much faster than in iteration 2. Below in *figure 3b* we can see a sample of the results. While there were less matches, the matches had a similar rate of accuracy as in iteration 2, at a comparatively much faster running speed.

PE090001	['cooking',	'equipment',	'industry',	'restaurant']	was	matched	with	48101820		Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
PE090002	['equipment'	'holding',	'industry',	'restaurant']	was	matched	with	20123004		Mining and Well Drilling Machinery and Accessories	Multilateral packer parts and accessories
PE090003	['equipment'	'industry',	'preservatior	'restaurant']	was	matched	with	45131507		Printing and Photographic and Audio and Visual Equipment and Supplies	Processed microfilm
PE090004	['equipment'	'industry',	'restaurant']	was	matched	with	48101820			Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
PE090005	['beverage',	'equipment',	'industry',	'restaurant']	was	matched	with	23181805		Industrial Manufacturing and Processing Machinery and Accessories	Ice making machine parts and accessories
PE090006	['concession'	'equipment',	'industry',	'restaurant']	was	matched	with	-1			
PE090007	['equipment'	'food',	'industry',	'preparation'	'restaurant']	was	matched	with	nan		
PE090008	['industry',	'refrigeratior	'restaurant']	was	matched	with	10161567			Live Plant and Animal Material and Accessories and Supplies	Alamo carolino tree
PE090009	['equipment'	'industry',	'restaurant',	'warewash']	was	matched	with	48101820		Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
PE090010	['industry',	'machines',	'restaurant']	was	matched	with	48111306			Service Industry Machinery and Equipment and Supplies	Restaurant customer management system
PE090100	['industry',	'restaurant',	'smallwares'	was	matched	with	48101820			Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
PE090101	['dinnerware	'industry',	'restaurant']	was	matched	with	48101901			Service Industry Machinery and Equipment and Supplies	Food service dinnerware
PE090102	['glassware',	'industry',	'restaurant']	was	matched	with	41101708			Laboratory and Measuring and Observing and Testing Equipment	Laboratory grinder or polisher
PE090103	['flatware',	'industry',	'restaurant']	was	matched	with	48101902			Service Industry Machinery and Equipment and Supplies	Food service flatware

figure 3b: Sample results for iteration 3, version 1 (titles and descriptions)

Meanwhile, running the program using only UNSPSC titles and no definitions in the input, the matching process took around 0.2 seconds per item for a total of 8.2 minutes, yielding 1666 matches out of 2544, with an accuracy of about 14% (results_sorted_2.txt). We see in *figure 3c* that similarly to the last iteration when we used only the titles, matches tend to be lower. We also had about 500 items that matched to 'nan' cells, which accounts for the lower number of matches. However, efficiency was vastly improved for slightly better results than in iteration 3.

P0000000	['personal',	'property',	'tangible']	was	matched	with	64121510					Financial Instruments, Products, Contracts and Agreements	Personal automobile insurance policy
P9999999	['default',	'taxable',	'temporary',	'unmapped']	was	matched	with	-1					
PA020000	['agricultural	was	matched	with	10171614							Live Plant and Animal Material and Accessories and Supplies	Potassium chloride for agricultural use
PA020100	['livestock']	was	matched	with	nan								
PA020111	['agricultural	'annual',	'commercial	'food',	'plants',	'producing']	was	matched	with	10171614		Live Plant and Animal Material and Accessories and Supplies	Potassium chloride for agricultural use
PA020113	['agricultural	'commercial	'food',	'producing',	'seeds']	was	matched	with	25101901				
PA020738	['animals',	'farm',	'food',	'game',	'wild']	was	matched	with	nan				
PA028802	['flowers']	was	matched	with	10161907								
PA020120	['agricultural	'commercial	'equipment',	'livestock',	'testing']	was	matched	with	25101901			Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020121	['agricultural	'cleaning',	'commercial	'equipment',	'livestock',	'supplies']	was	matched	with	nan			
PA020122	['agricultural	'commercial	'equipment',	'livestock']	was	matched	with	25101901				Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020123	['agricultural	'commercial	'equipment',	'livestock',	'vaccine']	was	matched	with	25101901			Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020225	['agricultural	'commercial	'grooming',	'livestock',	'supplies']	was	matched	with	25101901			Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020226	['agricultural	'commercial	'hygiene',	'livestock',	'supplies']	was	matched	with	25101901			Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020227	['agricultural	'commercial	'herbicides',	'insecticides'	'livestock',	'pesticides']	was	matched	with	25101901		Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020228	['agricultural	'commercial	'identificatio	'livestock']	was	matched	with	25101901				Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020229	['agricultural	'commercial	'feed',	'livestock']	was	matched	with	25101901				Commercial and Military and Private Vehicles and their Accessories and Components	Agricultural tractors
PA020230	['agricultural	'commercial	'livestock',	'safety',	'supplies']	was	matched	with	25101901				
PA020231	['agricultural	'commercial	'litter',	'livestock']	was	matched	with	25101901					
PA020300	['agricultural	'commercial	'livestock',	'medicine']	was	matched	with	25101901					
PA020301	['agricultural	'commercial	'counter',	'livestock',	'medicine',	'over',	'supplement	was	matched	with	25101901		
PA020302	['agricultural	'commercial	'livestock',	'medicine',	'prescription	was	matched	with	25101901				
PA020303	['agricultural	'commercial	'livestock',	'medicine',	'prescription	'sold',	'veterinarian	was	matched	with	25101901		
PA020304	['agricultural	'commercial	'livestock',	'sold',	'vaccines',	'veterinarian	was	matched	with	25101901			
PA020400	['agricultural	'commercial	'livestock',	'medical',	'supplies']	was	matched	with	25101901				
PA020401	['agricultural	'commercial	'livestock',	'stabilizers',	'vaccine']	was	matched	with	25101901				
PA020402	['agricultural	'commercial	'equipment',	'livestock',	'needles',	'related',	'syringes']	was	matched	with	25101901		
PA020003	['fertilizer',	'food',	'plant',	'plants',	'producing',	'root',	'tone']	was	matched	with	10171507	Live Plant and Animal Material and Accessories and Supplies	Urea fertilizer
PA020659	['food',	'fruit',	'herb',	'producing',	'seeds',	'vegetable']	was	matched	with	nan			
PA020661	['bulbs',	'fruit',	'onions',	'potatoes',	'vegetable']	was	matched	with	10152001			Live Plant and Animal Material and Accessories and Supplies	Fruit tree seeds or cuttings
PA020662	['food',	'plants',	'producing',	'woody']	was	matched	with	nan					
PA020664	['food',	'fungicides',	'insecticides'	'plants',	'producing']	was	matched	with	10111304			Live Plant and Animal Material and Accessories and Supplies	Pet food bowls or equipment

figure 3c: Sample results for iteration 3, version 2 (titles only)

Iteration 4: Sorted Inputs (Final)

Our final version of the program (see the /iter4 folder) is a refinement on iteration 3, taking better advantage of the nature of our word lists being sorted. We sought to retain the relative efficiency of the sorted inputs methodology while making every possible comparison, instead of skipping viable candidates as we did in iteration 3. As seen in *figure 4a*, we changed the conditionals within the innermost loop, such that we have three cases. If the Avalon word matches with the UNSPSC word, then we add one to the counter and attempt to match the next Avalon word with the next UNSPSC word. If the Avalon word is lexicographically larger than the UNSPSC word, then we compare the same Avalon word with the next UNSPSC word. If the Avalon word is lexicographically smaller than the UNSPSC word, then we compare the next Avalon word with the same UNSPSC word. Since the word lists are sorted, we know that if the current Avalon word is larger than a

UNSPSC word, then any Avalon word after itself must also be larger than that word. Likewise, if it is smaller than a UNSPSC word, then it is larger than any UNSPSC word that comes after. Therefore, we use the relative positioning of the words within their own lists to drastically reduce the number of comparisons. The main loop for each UNSPSC item ends when either list ends. At worst, we make the number of comparisons equivalent to the length of either word list, whichever is greater. Therefore, the complexity of comparing two word lists becomes $O(n)$. Even considering the linearithmic cost of sorting those lists, the result is faster than the previous $O(n^2)$ complexity.

```
ITERATION 4

def sortedMatch():

    for ava_item in Avalara_dict:

        max_count = 0, max_item = None

        for unspsc_item in UNSPSC_dict():

            while ava_index <= ava_lastindex and unspsc_index <= unspsc_lastindex

                while ava_index <= ava_lastindex:
                    while unspsc_index <= unspsc_lastindex:

                        if ava_item.wordlist[ava_index] == unspsc_item[unspsc_index]:
                            curr_count += 1
                            break

                        elif ava_item.wordlist[ava_index] > unspsc_item[unspsc_index]:
                            unspsc_index += 1

                        else:
                            break

                    ava_index += 1

            if curr_count > max_count:
                max_count = curr_count
                max_item = unspsc_item
```

figure 4a: Pseudocode for iteration 4

This approach vastly improved the accuracy of iteration 3, while maintaining the relative speed of its methodology. Since we were confident about the efficiency of the program, we ran it with the full titles and descriptions of the UNSPSC items at every level, omitting words smaller than 4 characters long. This yielded 2535 matches out of 2544, with an accuracy of about 30% (results_final.txt). The matching process took 59.3 minutes on a 2.3 GHz 8-Core Intel Core i9 processor (MacBook Pro 2019). Below in *figures 4b* and *4c* we can see two samples of the results, which improved on the accuracy of the same samples in previous iterations, and also provided new matches for those items previously matched to 'nan.'

596	PE090001 ['cooking', 'equipment', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
597	PE090002 ['equipment', 'holding', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
598	PE090003 ['equipment', 'industry', 'preservation', 'restaurant'] was matched with 45131507	Printing and Photographic and Audio and Visual Equipment and Su	Processed microfilm
599	PE090004 ['equipment', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
600	PE090005 ['beverage', 'equipment', 'industry', 'restaurant'] was matched with 23181805	Industrial Manufacturing and Processing Machinery and Accessori	Ice making machine parts and access
601	PE090006 ['concession', 'equipment', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
602	PE090007 ['equipment', 'food', 'industry', 'preparation', 'restaurant'] was matched with 48101601	Service Industry Machinery and Equipment and Supplies	Commercial use blenders
603	PE090008 ['industry', 'refrigeration', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
604	PE090009 ['equipment', 'industry', 'restaurant', 'warewash'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
605	PE090010 ['industry', 'machines', 'restaurant'] was matched with 48111306	Restaurant customer management system	Restaurant customer management s
606	PE090100 ['industry', 'restaurant', 'smallwares'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
607	PE090101 ['dinnerware', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
608	PE090102 ['glassware', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood
609	PE090103 ['flatware', 'industry', 'restaurant'] was matched with 48101820	Service Industry Machinery and Equipment and Supplies	Commercial kitchen hood

figure 4b: Sample results for iteration 4 – service industry machinery

1948	PB0010103 ['alarm', 'clock', 'consumer', 'electronics', 'embedded', 'lamp', 'portable', 'product', 'radio', 'radios', 'with'] was matched with 52161507	Domestic Appliances and Supplies and Consumer Electronic Products	Clock radios
1949	PB0010104 ['band', 'citizen', 'consumer', 'electronics', 'portable', 'product', 'radio', 'radios'] was matched with 52161555	Domestic Appliances and Supplies and Consumer Electronic Products	Portable video multimedia combined set
1950	PB0010105 ['consumer', 'electronics', 'gmrs', 'portable', 'product', 'radios', 'talkies', 'walkie'] was matched with 43191510		
1951	PB0010106 ['consumer', 'electronics', 'personal', 'portable', 'product', 'radios', 'transmitter'] was matched with 43191510		
1952	PB0010107 ['audio', 'consumer', 'electronics', 'monitor', 'portable', 'product', 'radios'] was matched with 52161555		
1953	PB0010108 ['consumer', 'electronics', 'portable', 'product', 'radio', 'radios', 'satellite'] was matched with 52161555		
1954	PB0010200 ['consumer', 'devices', 'electronics', 'playback', 'portable', 'product', 'recording'] was matched with 52161543		
1955	PB0010201 ['consumer', 'devices', 'electronics', 'playback', 'portable', 'product', 'recording', 'stereos'] was matched with 52161543		
1956	PB0010202 ['consumer', 'devices', 'docking', 'electronics', 'playback', 'portable', 'product', 'recording', 'stations'] was matched with 52161543	Domestic Appliances and Supplies and Consumer Electronic Products	MP3 players or recorders
1957	PB0010203 ['consumer', 'devices', 'electronics', 'media', 'physical', 'playback', 'player', 'portable', 'product', 'recorder', 'recording'] was matched with 52161560	Domestic Appliances and Supplies and Consumer Electronic Products	Home cinema system
1958	PB0010204 ['consumer', 'devices', 'digital', 'electronics', 'media', 'playback', 'player', 'portable', 'product', 'recording'] was matched with 52161605	Domestic Appliances and Supplies and Consumer Electronic Products	Portable media player accessories
1959	PB0010205 ['consumer', 'devices', 'electronics', 'headphones', 'playback', 'portable', 'product', 'recording'] was matched with 52161543	Domestic Appliances and Supplies and Consumer Electronic Products	MP3 players or recorders
1960	PB0010206 ['amplifier', 'consumer', 'devices', 'electronics', 'playback', 'portable', 'product', 'recording'] was matched with 52161543	Domestic Appliances and Supplies and Consumer Electronic Products	MP3 players or recorders
1961	PB0010207 ['consumer', 'devices', 'electronics', 'microphone', 'playback', 'portable', 'product', 'recording'] was matched with 52161543	Domestic Appliances and Supplies and Consumer Electronic Products	MP3 players or recorders
1962	PB0010300 ['cameras', 'consumer', 'electronics', 'equipment', 'photographic', 'portable', 'product'] was matched with 45101520	Printing and Photographic and Audio and Visual Equipment and Supplies	Industrial sign and label portable printer
1963	PB0010301 ['camera', 'cameras', 'consumer', 'electronics', 'equipment', 'photographic', 'portable', 'product'] was matched with 45121624	Printing and Photographic and Audio and Visual Equipment and Supplies	Photography light reflector
1964	PB0010302 ['camera', 'cameras', 'consumer', 'electronics', 'equipment', 'lenses', 'photographic', 'portable', 'product'] was matched with 45121624	Printing and Photographic and Audio and Visual Equipment and Supplies	Photography light reflector
1965	PB0010303 ['camera', 'cameras', 'consumer', 'electronics', 'equipment', 'flashes', 'photographic', 'portable', 'product'] was matched with 45121601	Printing and Photographic and Audio and Visual Equipment and Supplies	Camera flashes or lighting
1966	PB0010304 ['camera', 'cameras', 'consumer', 'electronics', 'equipment', 'photographic', 'portable', 'product', 'video'] was matched with 45121624	Printing and Photographic and Audio and Visual Equipment and Supplies	Photography light reflector
1967	PB0010305 ['cameras', 'consumer', 'electronics', 'equipment', 'photographic', 'portable', 'product', 'webcam'] was matched with 45101520	Printing and Photographic and Audio and Visual Equipment and Supplies	Industrial sign and label portable printer
1968	PB0010306 ['cameras', 'consumer', 'electronics', 'equipment', 'monitoring', 'photographic', 'portable', 'product', 'system', 'video'] was matched with 45101520	Printing and Photographic and Audio and Visual Equipment and Supplies	Industrial sign and label portable printer
1969	PB0010400 ['consumer', 'electronics', 'portable', 'printers', 'product', 'scanners'] was matched with 43211722	Information Technology Broadcasting and Telecommunications	Business card scanner

figure 4c: Sample results for iteration 4 – consumer electronics

Only 9 out of 2544 Avalara tax codes have no matches – a vast improvement over the 652 non-matches in iteration 3 version 1. Furthermore, there is a 6% improvement in accuracy (24% to 30%) corresponding to the number of matches. We conclude that the 19 extra minutes it takes to run iteration 4 is a reasonable cost for a measurably improved result.

CONCLUSION

In our final iteration, wherein we pre-sorted our input word lists and used their relative lexicographical order to ignore impossible matches, we managed to generate 2535 matches out of 2544, around 30% of which were accurate matches. This process took 59.3 minutes on a 2.3 GHz 8-Core Intel Core i9 processor (MacBook Pro 2019), and represented a balance between the speed of the first iteration and the accuracy of the second iteration, using a methodology we developed in the third iteration.

Our work was split evenly among our team members throughout the weeks we spent on this project. Jason spearheaded the Rabin-Karp matchmaking and the many versions of the Avalara data parsing throughout the whole process, while Norrec led on the sorting solutions and the UNSPSC data parsing. Jason did a lot of data analysis and visualization on the results, while Norrec worked on developing strategies for parsing and representing the Avalara and UNSPSC data. We consistently checked each other's work, pair-programmed and implemented new functions independently, manually parsed the data files, and discussed possible solutions. We compiled the reports and gathered data on our results together.

Future improvements to the program could involve implementing a version of the matchmaking that depended on determining 'parent' and 'child' nodes in the Avalara file as we did in iteration 2, and using the matching results of the 'parent' node to narrow down the target candidates for its 'child' nodes to a class or family of UNSPSC items. This would vastly improve efficiency, but the effects on accuracy are difficult to predict, since a bad match on the parent could negatively affect the child, while a good match could positively affect the child. We would also attempt to introduce prioritizing of keywords in the UNSPSC word lists, and improve on the output format using pandas so as to reduce the manual work of parsing our results.

WORKS CITED

"Pandas Documentation." Pandas. Version 1.5.2. Nov 22, 2022. https://pandas.pydata.org/docs/

"The Python Standard Library." Python. Version 3.11.1. https://docs.python.org/3/library/index.html

"Rabin-Karp Algorithm for Pattern Searching." Geeksforgeeks. Sep 23, 2022. https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/