# OrgMatcher – Honors Contract Project

Lance Joseph A. Trasporto

College of Engineering, University of North Texas

Honors Contract

Dr. Xinrui Cui

January 17, 2026 (FIXME)

***Note that everything is still WIP, and I basically just wrote down what was on my mind. Expect many revisions in the near future**

## I. Introduction (WIP)

As of January 2026, there are over 400 different organizations that are active at UNT. For incoming freshmen or transfer students, this could be quite intimidating, even if there are many choices presented before them. How could they find organizations that they believe they can fit in with? How could they be sure these organizations align with their general interests and hobbies? When a student visits UNT's OrgSync page, they are provided with a search bar if they already have an organization in mind, as well as a "filter categories" section to look for organizations that align with their general interests. However, what about students who do not have a clear idea of what they want?

This is the problem that OrgMatcher tries to solve. OrgMatcher is a program that prompts students to specify their interests, hobbies, or beliefs, and matches them with the five most relevant organizations currently active at UNT. This program will aim to get more students involved with organizations on campus, as well as provide incoming students, whether they are freshmen or transfers, a smoother transition into UNT by providing them with a group they belong to.

This paper will cover the description, the development process of the program, and methodologies used, with emphasis on natural language processing techniques, the vision behind the program, as well as a discussion section on how the application can be considered a scholarly contribution and improved upon.

## II. Description (WIP)

OrgMatcher is a natural language processing (NLP) program that prompts a user for a string of input where they can provide the qualities they look for in an organization, which can range from their hobbies, interests, social connections, personal growth, and various other personal factors. The program returns the five most relevant organizations based on the user's input, which matches their input based on the words listed on the description pages of each respective organization on UNT's OrgSync website.

Over 400+ UNT organizations have been included in this program, and factors such as the organization's name, abbreviated name, summary, and description have been included.

**III. Process and Methodology (WIP)**

For this program to function, it first needs data to compare the user input with. This is done using the programming language Python, as well as a well-known library called BeautifulSoup. I created a script that automatically scrapes the links of all organizations through UNT's OrgSync API. These links are then stored in a .csv (comma-separated value) file. With this CSV file in hand, another script automatically accesses the links in the file line by line, parsing and scraping an organization's name, acronym (if available), summary, description, and logo.

The parsed, raw HTML data is then cleaned and stripped of any commas, links, colons, or semicolons, and is passed into a SQLite database for easier access to club information that will be used later in the program. From now on, we will refer to the data stored in the database as the corpus. A **corpus** is a collection of text that is used to study language use or train AI models (Egger & Gokce, 2022). In this scenario, we will use the corpus to analyze the term frequency and inverse document frequency for each word in the document.

Then begins the process of vectorization and TF-IDF, where Python libraries scikit-learn and pandas will be utilized to implement the natural language processing model of the program. This model will vectorize the user's input–a string of text–into a list of words that will then be used, compared, and given a "score" or weight through TF-IDF. Parameters such as the organization's name, short name (e.g., IEEE for Institute of Electrical and Electronics Engineers), summary, and description will also be combined and vectorized. From now on, we will refer to these combined parameters as the document. For this project, let us define a document with this equation:

document = organization's name + short name + summary + description

**Term frequency-inverse document frequency (TF-IDF)** is a widely used natural language processing technique that calculates a numerical weight for each word to reflect how important that word is to a specific document within a collection or corpus (Robertson, 2004). TF-IDF is actually a combination of two equations: Term frequency (TF) and Inverse Document

Frequency (IDF). These two equations are multiplied to give a word a "score" that determines a word's importance.

Term frequency measures how often a word appears in a document. It is defined with the following general equation:

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

The equation takes parameters *t* and *d*, where *t* is the number of times a word appears in document *d*. For this example, we use the organization Asian Student Association. As mentioned above, the document consists of the organization's name, short name, summary, and description. It is also good to note that Python's scikit-learn library converts all words to lowercase, strips all punctuation, and disregards single-letter words such as 'I' and 'a' during vectorization, giving us the following:

[ 'asian', 'student', 'association', 'asa', 'group', 'striving', 'to', 'promote', 'asian', 'culture', 'through', 'informative', 'meetings', 'intramural', 'sports', 'and', … ]

Using this information, let us apply the word 'Asian' to the text frequency formula using the vectorized document above. For context, the word 'Asian' appears seven times, and the word 'Asians' appears once. The program will count these words similarly, making the base word 'Asian' appear eight times, through a process called lemmatization. **Lemmatization** is a technique used to reduce a word to its base or root form (Pant et al., 2024). For example, words such as 'runs', 'running', and 'ran' will all be reduced to 'run'. In this scenario, 'Asians' is reduced to 'Asian'. We get the following result:

$$TF = \frac{\text{Number of times 'Asian' appears in document}}{\text{Total number of terms in document}} = \frac{8}{86} \approx 0.0930$$

The term frequency formula thus gives the word 'Asian' a score of around 0.0930.

On the other hand, inverse document frequency measures how rare a given word is across all 400+ documents. It is defined with the following general equation:

$$IDF(t, D) = \log_e \left( \frac{\text{Total number of documents in corpus}}{1 + \text{Number of documents with term } t \text{ in them}} \right) = \log_e \left( \frac{D}{1 + df(t)} \right)$$

The equation takes parameters *t* and *D*, where *t* is the term, and *D* is the corpus. The variable *df* is the document frequency of a term. **Document frequency** measures how many unique documents contain a specified term (Joho & Sanderson, 2007).

Let us again use the word 'Asian' as an example. The term 'Asian' has a document frequency of 8, meaning across the whole corpus of 416 documents in the corpus, the term 'Asian' appeared in 8 of them. Plugging this into our equation, we get:

$$IDF = \log_e(\frac{416}{1+8}) \approx 3.8335$$

The inverse document frequency thus gives the term 'Asian' a score of around 3.8335. To get the combined TF-IDF weight, we multiply the results from the TF and the IDF equations.

$$TF - IDF(t, d, D) = TF(t, d) \cdot IDF(t, D) = 0.0930 \cdot 3.8335 \approx 0.3565$$

For comparison, let us apply the same logic using a more common word, such as 'the'. The word 'the' appears on 371 out of the 416 documents in the corpus. Below, variable *d* is the document for the Asian Student Association, and *D* is the corpus.

$$\begin{aligned} TF(\text{'the'}, d) &= \frac{3}{86} \approx 0.0349 \\ IDF(\text{'the'}, D) &= \log_e \left( \frac{416}{1 + 371} \right) \approx 0.1118 \\ TF - IDF(\text{'the'}, d, D) &\approx 0.0039 \end{aligned}$$

The word 'Asian' weighs around 0.3565, while the word 'the' weighs around 0.0039. This reveals that the word 'Asian' is approximately 91 times more significant in the corpus, meaning that when a student enters the word 'Asian' into the program, it will be considered a word with

more value compared to more common words, such as ‘the’. Since the word ‘the’ is commonly occurring and not heavily context-dependent, we will add it to a list of stop words and disregard it. **Stop words** are common words (like ‘the’, ‘is’, ‘of’, etc) that have little semantic meaning and are filtered out for a more accurate analysis of terms (Wilbur & Sirotkin, 1992). Additionally, we will give short name inputs an extra 2* weight, since we assume that if a student includes the name of an organization in an input, then there’s already an initial interest.

Then, the student’s input will be compared with the documents we used TF-IDF on through a method called cosine similarity. **Cosine similarity** is a commonly used technique in natural language processing, where it measures similarity between text documents based on their term vectors (Rahutomo et al., 2012). Two completely similar vectors will have a cosine similarity of 1, two completely unsimilar vectors will have a cosine similarity of 0, and vectors with some similar terms will fall between the range 0 and 1. The closer the number is to 1, the higher the similarity. It is defined with the following general equation:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where variables $A$ and $B$ are vectors of terms and $A_i$ and $B_i$ are elements at the $i^{\text{th}}$ position, $i$ being the first or initial word in the vector, and $n$ being the length of the vector.
For a quick example, let us consider the following sentences. Remember that single-letter words and punctuation will be stripped, and all words will be converted to lowercase.

A. “UNT is a great university for students!” → [‘unt’, ‘great’, ‘university’, ‘students’]
B. “I love UNT and the campus.” → [‘love’, ‘unt’, ‘campus’]
C. “Hi there.” → [‘hi’, ‘there’]
D. “The UNT campus is great, and I love it!” → [‘unt’, ‘campus’, ‘great’, ‘love’]

With this information, let’s create a frequency table for the terms in the vector and remove the stop words. The second row will be for sentence A, the third row for sentence B, and so forth.

| ‘unt’ | ‘great’ | ‘university’ | ‘students’ | ‘campus’ | ‘love’ | ‘hi’ | ‘there’ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

Now, we can substitute these numbers directly into the formula.
Using the formula on sentences A and B, we get:

$$\text{Cosine Similarity}(A, B) = \frac{(1 \cdot 1) + (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 1) + (0 \cdot 0) + (0 \cdot 0)}{\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2 + 0^2 + 0^2}\sqrt{1^2 + 0^2 + 0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 0^2}} \approx 0.2887$$

Using the formula on sentences A and C, we get:

$$\text{Cosine Similarity}(A, C) = \frac{(1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 1) + (0 \cdot 1)}{\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2 + 0^2 + 0^2}\sqrt{0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 1^2}} = 0$$

Using the formula on sentences B and D, we get:

$$\text{Cosine Similarity}(B, D) = \frac{(1 \cdot 1) + (0 \cdot 1) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 1) + (0 \cdot 0) + (0 \cdot 0)}{\sqrt{1^2 + 0^2 + 0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 0^2}\sqrt{1^2 + 1^2 + 0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 0^2}} \approx 0.8660$$

As observed, sentences B and D are most alike, with a cosine similarity of around 0.8660, which is evident based on the frequency table above. On the other hand, we can observe that sentences A and C are completely dissimilar.

In the actual program, this process will be done on all 416 documents in the corpus, as well as for the user's input string. The TF-IDF and cosine similarity score will be multiplied, and the program will return and display to the user the first 5 organizations that have the highest score and have the most relevance based on the user's input. While these methodologies have been greatly abstracted by the Python libraries mentioned above, such as scikit-learn and pandas, utilizing libraries is a common approach and is considered a standard that are in use and have real-world impact

Lastly, the program is given a user-friendly interface by utilizing the frameworks React for the frontend and FastAPI for the backend (**FIXME**)

**Example Case:**

Let the student's input be:

"I'm an Asian transfer student looking to make new friends. I'm also interested in clubs that have some kind of physical activity involved."

- **(FIXME)**

## IV. Vision (WIP)

The vision for this program is mainly for freshmen or transfer students to have a more friendly and easier transition into UNT by being a part of an organization during their time here (add extra words here). For me personally, I was overwhelmed with the number of organizations here at UNT, and I believe this program could narrow down another student's choices when it comes to which organizations to be involved in.

I believe it would also be of benefit to UNT if a program such as OrgMatcher, or a program that functions like it, were implemented into UNT OrgSync's website. I believe that a simple addition–a text box prompting a student what their interests are–would undoubtedly help students get exposed to the breadth of different organizations here at UNT.

I also believe that this application does not just have to be UNT organizations; it could be modified and used to notify students of upcoming events, meetings, career fairs, and other happenings at UNT.

## V. Discussion (WIP)

Compared to the current version at the OrgSync website that uses a simple search bar and a category filter, I would argue that this is a greater upgrade in comparison. As mentioned above, a student has to know what they are looking for when they type in a search bar, whereas this NLP application will compare the student's interests with the description listed on the organization's website.

A data constraint to this program is that it does not parse the category of an organization. Based on UNT OrgSync's website, all organizations have their "categories" header as empty when looking at the source code. An option would be to manually add categories for each

organization, but this would be highly impractical, and it would require manual updates if there are newer organizations are created in the future.

Another limitation of this program would be that TF-IDF does not understand context and human intentions. This approach could be handled using transformers that are specifically trained to learn about the 400+ organizations at UNT. Though I would argue that this current implementation is good enough and does the job quite well already.

Another great addition to this program would be implementing sentiment analysis along with the already-existing TF-IDF to increase the accuracy of matching a student based on the organization's description, based on the organization's "vibes" (**FIXME**).

Lastly, I assert that this application is a scholarly contribution. OrgMatcher is a testament to the advancement of AI, utilizing machine learning practices to solve a practical social problem on campus that efficiently connects students to the organizations where they can build their community and thrive in a new environment. (**FIXME**)

## References

Egger, Roman & Gokce, Enes. (2022). Natural Language Processing (NLP): An Introduction: Making Sense of Textual Data. 10.1007/978-3-030-88389-8_15.

Joho, Hideo & Sanderson, Mark. (2007). Document Frequency and Term Specificity.

Pant, Vinay & Sharma, Rupak & Kundu, Shakti. (2024). An overview of Stemming and Lemmatization Techniques. 10.1201/9781003430421-31.

Rahutomo, Faisal & Kitasuka, Teruaki & Aritsugi, Masayoshi. (2012). Semantic Cosine Similarity.

Robertson, Stephen. (2004). Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. Journal of Documentation - J DOC. 60. 503-520. 10.1108/00220410410560582.

Wilbur, W. & Sirotkin, Karl. (1992). The automatic identification of stop words. Journal of Information Science. 18. 45-55. 10.1177/016555159201800106.