A DATA-DRIVEN APPLICATION FOR MATCHING STUDENT TRAITS WITH LEARNING OPPORTUNITIES USING ARTIFICIAL INTELLIGENCE

Chongda You[1], Andrew Park[2]

[1]Portola High School, 1001 Cadence, Irvine, CA 92618
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

It is often limited to students the opportunities they can get to apply their knowledge and learn new things, as different opportunities have vastly different ways of advertising. Just like the students, many organizations are looking for passionate students to apply their knowledge and energy to benefit society [2]. To solve this problem and ease the difficulties in finding the right opportunities, Maclever aims to be a place where all organizations can post their opportunities for students, and students can use the artificial intelligence-based feature in this application to find opportunities that best fit their skills. Maclever aims to be a simple and effective connection between organizations and students [3]. Leveraging tools such as sentiment analysis and utilizing models based on user behaviors and preferences to better match valuable connections allows us to create a system that gives us a much stronger ability to address the goals Maclevers sets out to solve.

KEYWORDS

Data-Driven, Sentiment Analysis and Models, User Interaction, Sentiment Analysis Model

1. INTRODUCTION

It is often limited to students the opportunities they can get to apply their knowledge and learn new things, as different opportunities have vastly different ways of advertising. Just like the students, many organizations are looking for passionate students to apply their knowledge and energy to benefit society. It would be painful to see that good organizations cannot find good students and vice versa. Having the right opportunities to learn is crucial for students to expand their knowledge in the fields they are interested in, therefore it should not be hard to find.

To solve this problem and ease the difficulties in finding the right opportunities, Maclever aims to be a place where all organizations can post their opportunities for students, and students can use the artificial intelligence-based feature in this application to find opportunities that best fit their skills [4]. Maclever aims to be a simple and effective connection between organizations and students.

To shape the automatic recommendation system for Maclever, we will be leveraging tools such as sentiment analysis and utilizing models based on user behaviors and preferences to better match valuable connections.

David C. Wyld et al. (Eds): ARIN, CSITA, ISPR, NBIOT, DMAP, MLCL, CONEDU -2023
pp. 157-168, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130714

There are very few existing methods for addressing the problem of matching students and organizations in an easy-to-use system. [insert brief description about what those other methodologies accomplish] With that being said, these methods do not quite accomplish these purposes in a way that is all unified within one app to make the overall experience from start to finish more streamlined. From a more general sense, there are other additional methods employed by other entities to further strengthen the results of sentiment analysis such as "NLP techniques (Bag-of-Words and TF-IDF) and various ML classification algorithms" which can be used to "find an effective approach for Sentiment Analysis on a large, imbalanced, and multi-classed dataset"[12].

Our application sets out to utilize user usage data and sentiment analysis to better address the needs of the students and provide better connections between them and the creators seeking to reach them. We do this through a separation of concerns, a sentiment analysis engine tailored to each user, and a combination of direct and implied user patterns to continue evolving with the needs of the users [5]. In terms of separation of concerns, the app supports two different account types with very specific abilities and features. Student accounts can read and respond to posts as well as receive proper recommendations. Creator accounts, on the other hand, are not able to see any other posts besides their own along with handling responses from student accounts towards the posts they created. This means that usage of the system is more purposeful in its objective in a way not many other apps can do. To shape the automatic recommendation system for Maclever, we will be leveraging tools such as the dart sentiment library, which normally uses AFINN-165 "a wordlist-based approach for sentiment analysis", and Emoji Sentiment Ranking to perform sentiment analysis based on a given text input [13]. This sentiment model is then further adjusted by tracking what posts students are responding to alongside any explicit preferences they make along the way. The idea for this application was sparked by a recognition from the perspective of a high school student that the lack of this kind of tool is a very prevalent issue. With all of these aspects combined, the app will be able to serve a wide range of purposes and topics that streamlines making connections.

Within Section 4, we conducted a series of experiments to test the efficacy of our methods and implementations. For the first experiment, we focused mainly on the accuracy of the user experience and how the recommendation matched up to the expectations of the end user. To do this, we constructed a set of user profiles catered to various fields or occupations and assessed the accuracy of the resulting "For You" page's post order. The main conclusion from the end of this was that there is currently a STEM bias that is caused in large part due to their overrepresentation in the recommendation system and the current weights attached to each of these [6].

The second experiment was more focused on the general engagement of users on Maclever and tested what factors lead to a higher click rate alongside what users perceived to be additional improvements they want beyond. We tracked the click rate of various posts on the same topic but with varying formats to determine how they affect the overall performance of a given post. We then followed up with a general survey to gather what features users will be looking to have added to Maclever, much of which was centralized around improvements to the communication channels and the flexibility of the tags system to better cater to the recommendation engine.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Distinguish the Author of Each Post

A major component of Maclever is having the creator version home page showing what posts they have created because all posts were saved in one Firebase collection with no way of telling which is created by which [8]. To distinguish the author of each post, I could have another Firebase collection named user_data to store all the posts created by each creator, meaning each creator account has a document in this collection with all his posts in it. In the creator account homepage, all the posts created by that creator could then be abstracted from this document.

2.2. Efficient Read/Write Structure

In regards to the backend database, there must be an efficient read/write structure in place to minimize database operation costs. Backend-as-a-Service platforms such as Firebase will often have free tiers followed by scaling costs to determine how much should be charged for database changes. To minimize the operating costs given such limitations, there are a variety of methods such as caching. Caching is the act of keeping a copy of user-relevant data on the client side and using that instead of reading from the database every time an operation should occur. By doing it this way, there is no need for performing a read every single time to the backend and instead only doing so when a change is directed via listeners [9]. The actual execution of this caching behavior can be done through runtime-specific tools such as singletons to keep information in memory or through local save data that can persist between sessions.

2.3. Preference Recommendation

The most important feature for students is its preference recommendation, which is how activities should be recommended to students with different knowledge, interests, and skill sets. The solution to this challenge would include two parts: how to get students' preferences, and how to recommend activities according to these preferences. To get students; preferences, there could be a short quiz that would pop up the first time a student clicks on the "For You" screen, which includes several simple questions that focus on what the student's main focus and academic goal is. The results of this quiz could be used to set up parameters for the student's "preference tags", which are numbers assigned to tags such as "English", "volunteer", and "Business". In the second part of the solution, these parameters can then be used in a sentiment intelligence model to put activity posts with matching tags on the "For You" screen for a student.

3. SOLUTION

Maclever is written in Flutter and based on Firebase. Flutter is used for the programming part, while Firebase stores all the user account information ranging from post history for creators to the personalized recommendation models for student accounts and all posts made by creator accounts.

A brief flow of Maclever is: a user can create either a student or creator account at the beginning. A student account can see all the posts about learning opportunities created by creator accounts, or they can let Maclever's AI-based recommendation system filter out activities that have the best matching features to the students' preferences, which is given to the AI after the student takes a short quiz [10]. A creator account is used for seeking students for the available activities by creating posts, it can see and edit all activity posts created by that creator, and create new posts.

All posts created by different creators will enter the pool of posts that show up on the student accounts.

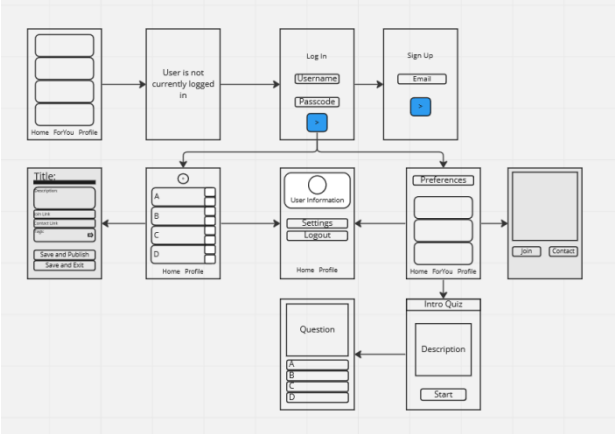


Figure 1. Overview of the solution

One of the primary goals of this application is to help students find new opportunities and make more connections. To that end, our system must have a specific account type that is tailored to this kind of need [11]. Student accounts have dedicated access to the recommendation engine in addition to the ability to read all the created posts. This focus on a particular feature is enabled for such accounts which are differentiated on the Firebase backend.

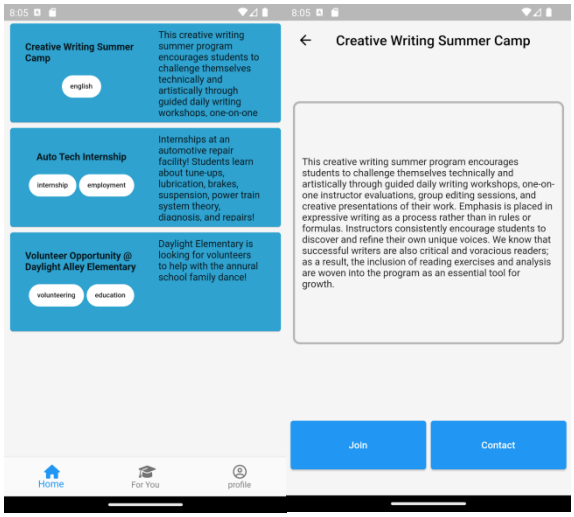
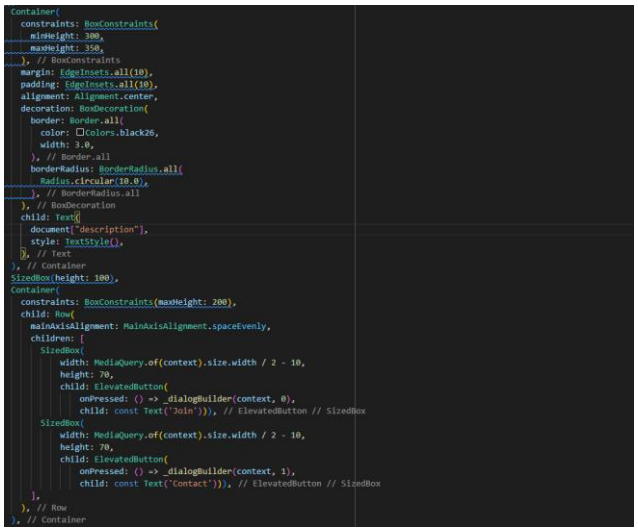


Figure 2. Screenshot of the app 1



```
Container(
  constraints: BoxConstraints(
    minHeight: 300,
    maxHeight: 350,
  ), // BoxConstraints
  margin: EdgeInsets.all(10),
  padding: EdgeInsets.all(10),
  alignment: Alignment.center,
  decoration: BoxDecoration(
    border: Border.all(
      color: Colors.black26,
      width: 3.0,
    ), // Border.all
    borderRadius: BorderRadius.all(
      Radius.circular(10.0),
    ), // BorderRadius.all
  ), // BoxDecoration
  child: Text(
    document["description"],
    style: TextStyle(),
  ), // Text
), // Container
SizedBox(height: 100),
Container(
  constraints: BoxConstraints(maxHeight: 200),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      SizedBox(
          width: MediaQuery.of(context).size.width / 2 - 10,
          height: 70,
          child: ElevatedButton(
              onPressed: () => _dialogBuilder(context, 0),
              child: const Text('Join')), // ElevatedButton // SizedBox
      SizedBox(
          width: MediaQuery.of(context).size.width / 2 - 10,
          height: 70,
          child: ElevatedButton(
              onPressed: () => _dialogBuilder(context, 1),
              child: const Text('Contact')), // ElevatedButton // SizedBox
    ],
  ), // Row
), // Container
```

Figure 3. Screenshot of code 1

This particular snippet of code runs when a student taps on a particular post that they may find interesting and want to look up the information for. Said information for any given creator post is stored on the backend database where the student account can then retrieve it. Upon showing up in the feed, the metadata regarding that post is locally cached until it is needed for when the user taps on the post preview. The respective post is then rendered in a topics screen which includes the Container pictured above, which will allow the student to gain access to the post description alongside the necessary contact information.

On the other side of the coin, creators are also in need of finding students with whom they can connect to help foster new relationships and find new talent that they can work with. To serve that purpose, our app also includes a dedicated set of features for “Creator Accounts” which can create and edit posts but are not able to read the other posts to maintain their focus.

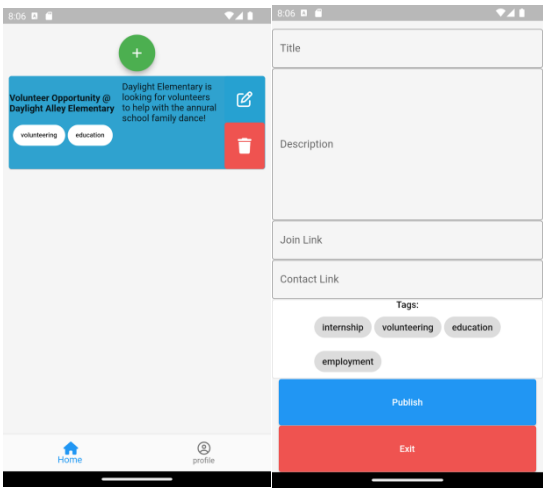


Figure 4. Screenshot of the app 2

```
onPressed: () {
  _singleton.editDirty = false;
  if (_singleton.currentDocument == null) {
    FirebaseFirestore.instance.collection('topics').add({
      'title': title,
      'description': description,
      'join_link': joinLink,
      'contact_link': contactLink,
      'author': Auth().user!.uid,
      'tags': selectedReportList,
    }).then((value) {
      // print(value.id);
      FirebaseFirestore.instance
          .collection('user_data')
          .doc(Auth().user!.uid)
          .update({
        "posts.${value.id}": {
          'title': title,
          'description': description,
          'join_link': joinLink,
          'contact_link': contactLink,
          'author': Auth().user!.uid,
          'tags': selectedReportList,
        }
      }).then((value) =>
              Navigator.pushNamed(context, '/homeCreator'));
    });
  } else {
    FirebaseFirestore.instance
        .collection('topics')
        .doc(_singleton.currentDocument!["doc_id"])
        .update({
      'title': title,
      'description': description,
      'join_link': joinLink,
      'contact_link': contactLink,
      'author': Auth().user!.uid,
      'tags': selectedReportList,
    }).then((value) {
      // print(_singleton.currentDocument);
      FirebaseFirestore.instance
          .collection('user_data')
          .doc(Auth().user!.uid)
          .update({
        "posts.${_singleton.currentDocument!["doc_id"]}": {
          'title': title,
          'description': description,
          'join_link': joinLink,
          'contact_link': contactLink,
          'author': Auth().user!.uid,
          'tags': selectedReportList,
        }
      }).then((value) {
        _singleton.currentDocument = null;
        Navigator.pushNamed(context, '/homeCreator');
      });
    });
  }
},
```

Figure 5. Screenshot of code 2

This snippet of code is from the “Create/Edit Posts” screen which is accessible from the creator homepage either through a green “+” button or through the existing posts they have (which is what will allow for editing and deleting of posts). If the user is editing one of their existing posts, then the screen will open up with all of the current info already filled in as opposed to empty which would be the case for creating a new post. Once the user is ready to publish their changes, the application will then run the code shown above where the information will be sent to both the public topic store to allow student readers to see it as well as a separate copy kept in the user’s data. The reasoning for this particular aspect is that it allows for easy tracking and caching of the posts made by the creator which ultimately also allows us to minimize necessary database reads to save on operation costs.

To tie these two account types and their respective functions together, we must also have a recommendation engine powering the interactions between the two to make sure that users are getting connected to the information that is best paired with what they are looking for. To help in that aspect, we are using a sentiment analysis strategy that is updated based on the user’s preferences and behavior.

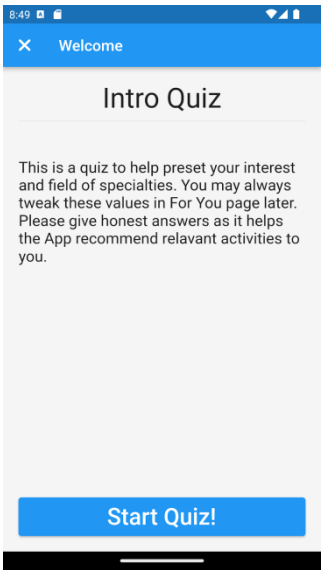


Figure 6. Screenshot of app 3

```
if (_singleton.postsCache != null) {
  sortedPosts = _singleton.postsCache!
      .map((topic) => topicEntry(
            document: topic,
          ))
      .toList();

  sortedPosts.sort(((a, b) => Sentiment.analysis(
          b.document["tags"].join(' '),
          customLang: customLang)
      .score
      .compareTo(Sentiment.analysis(a.document["tags"].join(' '),
              customLang: customLang)
          .score)));
}
```

Figure 7. Screenshot of code 3

The snippet of code above is a small part of the overall recommendation system that is used in Maclever. Once a student account can enter the “For You” screen, the posts that have been retrieved in the home screen are then processed through the recommendation engine. The “customLang” variable corresponds to the model shaped specifically for the user that will be used by the sentiment analysis system to calculate a score which is then used to rank the posts by order of interest and relevance. It should also be noted that this model creation is assisted by an introduction quiz that must be completed to unlock the “For You” screen. This allows us to form a rudimentary model right away that can then be improved down the line as general usage data is analyzed. We also have a singleton storing a local cache of the posts retrieved so that the general sorting and preference logic can be run locally to further optimize whatever information is provided by the backend. The caching behavior paired with the singleton pattern is also a great way to store information about posts already retrieved so that duplicated read requests do not need to be constantly performed which results in a more efficient application that incurs fewer operation costs without interfering with the user experience.

4. EXPERIMENT

4.1. Experiment 1

One of the primary concerns with the app is whether or not the recommendation system is accurate enough to properly identify what a given user would find more engaging and useful to their preferences and needs.

For this experiment we will create a series of profiles of different students with a set of pre-defined interests to emulate a user going into the application. We will then answer questions and behave in a manner matching each of these profiles then tracking the order of their recommended posts to see how it compares to the actual profile of the simulated user.

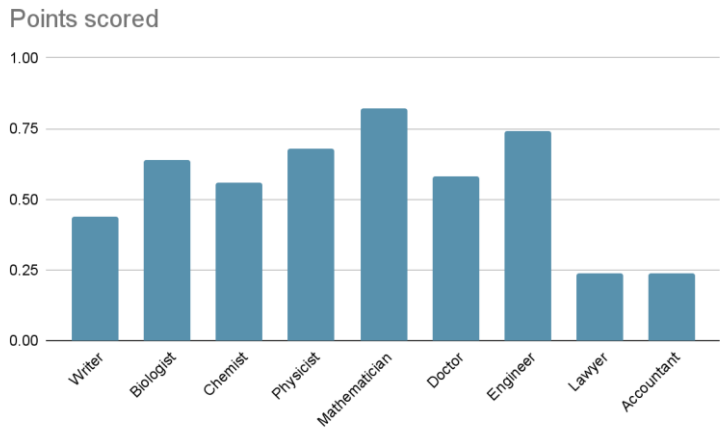


Figure 8. Points scored

For each one of the occupational profiles we constructed, a specific pool of tags was curated for each one and then matched alongside a group of posts that related to each one of these occupations. Each profile was then given an accuracy score based on the sequence of the posts as they show up on the "For You" screen. The calculation for a given post's score was calculated as (1.0 / relative placement) and then averaged out to give a cumulative score on how the recommendation system performs for that particular profile. A notable trend that was observed is that there was a higher accuracy rate for profiles that were related in some way to the STEM field. This ran counter to the initial hypothesis which was that more niche topics would be more pronounced as fewer similar topics need to compete, but the fact that this is the case is indicative of there currently being an overabundance of STEM-related topics in the recommendation system with weights not sufficiently being set to knock down less relevant topics for users who are interested in topics outside of STEM.

4.2. Experiment 2

An additional concern that may arise even if the recommendation engine is accurate is whether or not the underlying system itself is doing enough to make sure the right students are being connected to the right organizations.

A good way to identify immediate pain points that we can address is to track the engagement for a given post and see what aspects of that post are different compared to others. For the posts with

the most engagement, we can then survey what particular aspects users would want to add to further streamline the experience. We will be addressing these questions by testing the click rate of three different posts and following up with a general survey for users who viewed said posts.

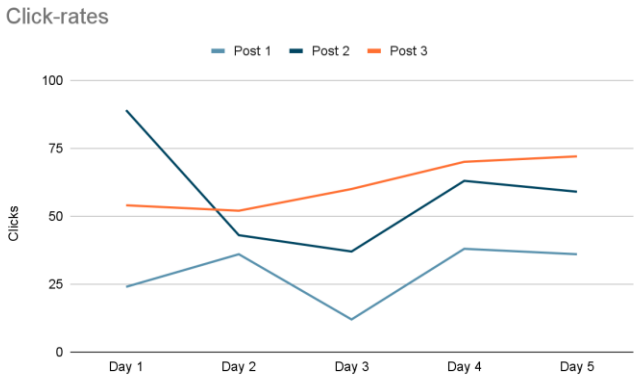


Figure 9. Click rates

For the general engagement test, we spent time tracking user activity over several days with each of the given posts holding key differences in their format. The first post started as our control with the overall post containing the following content:

Title: Writing Competition

Description: Are you a student looking for a platform to showcase your writing skills? Our…
Tag(s): writing

Join Link: https://www.competition.com
Contact Link: https://www.contact.com

The other two posts were essentially variants of the aforementioned post with the following changes:

Post 2: Title changed to "✍ Story Challenge ($500 prize)"

Post 3: title from post 2 and the addition of the tags "english", "story", "fiction"

Each one of the changes translated to an increasingly positive impact on the overall performance of the post compared to the one before, although how it does differs in some key ways. The control post had a generally small impact in terms of click rate with the occasional fluctuation over time. For the second post, on the other hand, there was a noticeable jump in clicks throughout the day initially on account of the revised title. The main observation from this is that the use of emojis and a clear incentive provided a visual cue that resulted in a significantly larger amount of attention. With that being said, a notable quirk of this post was that there was a significant drop soon after before remaining at a consistent spot somewhat above the metrics of post 1. Finally, for the third post, we incorporated more tags to try and cast a wider net over who might be interested in the post. In this case, the post resulted in a slightly lower click rate initially but was differentiated by a direct positive trend that led to it outperforming the previous posts by a significant margin.

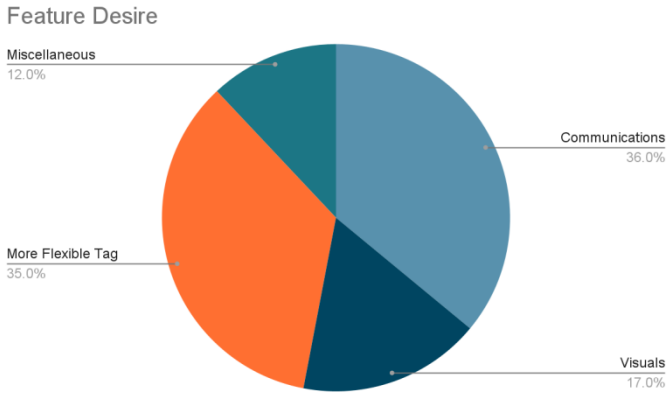


Figure 10. Feature desire

For those involved with the most engaging posts, we also took a general survey about what features they believe would make their experience better. The most significant highlighted feature was communication with a 36% share of user interest. This is in large part due to the communications between organizations and students having to be facilitated in large part through the sites and contacts listed on the post as opposed to an all-in-one solution such as a direct chat feature within the app itself. The second largest share was a request for a more flexible tag system, which falls in line with the observations made from our click-rate experiment as the ability to add a lot of relevant tags was directly correlated with boosted performance under our recommendation system. There was also a notable share of interest in visual tweaks which directly pertained to profile presentation, expressing the need for more social features for the app overall.

5. RELATED WORK

Researcher Jeremiah Chun has created a website that recommends college majors for high school students applying to colleges [14]. The recommended major is based on a quiz that asks about the student's classes taken in high school. This methodology is similar to the quiz on Maclever that uses a short quiz to identify the traits of a student, therefore being able to recommend activities with matching traits. Chun uses survey results from graduates about their past high school courses and selected majors to train a model that can be used to predict majors based on courses. This is a great solution for trying to find patterns in a set of data (selected courses) to match the pattern with a result (predicted major). Maclever also uses a quiz to predict what type of activities a student is interested in. However, since the quiz's output is multiple traits a student has, it is more efficient to have these traits as tags and let the quiz determine a corresponding point number to each tag. A larger number for one tag means the student is stronger in this trait(such as math, geography, and leadership). All these tag values can then be used to train a sentiment model which can take in the trait numbers of a student and find activities with matching tags.

Naviance is another piece of software that seeks out a similar mission to Maclever in that it promotes American college readiness among K-12 students. Their approach provides a wide range of functions such as college research tools, career and course planners, surveys, and personality tests to help the end user reflect on and identify things they might want to do to be better prepared in the future. To help them visualize the efficacy of their tools, they maintain scattergrams that show how student data is correlated with where they end up attending. It was found that average annual logins are one of the strongest predictors accounting for 89% followed

by the “Grade point average [which] was the second best predictor of college application rate explaining 48% of the effect” [7].

One particularly notable technique used beyond sentiment analysis is done by Uber and goes under the name of CSS (Contextual Semantic Search) to get said task done. This method seeks to address the issues of a conventional approach which filters all keyword-related information by searching the keyword and its related terms, namely the fact that it “is not very effective as it is almost impossible to think of all the relevant keywords and their variants that represent a particular concept” [1]. What is done instead is that each word is converted by the AI to a point within a larger hyperspace from which “the distance between these points is used to identify messages where the context is similar to the concept we are exploring” [1]. Techniques like CSS can be used in tandem with sentiment analysis to get a more complete picture of what the nature of the content being parsed actually will be.

6. CONCLUSIONS

There are several limitations and points for improvement that must be considered for Maclever. One particularly important aspect is making sure that the underlying systems in place can properly adapt to the changing behaviors of a given user to then adjust what it should recommend for a given student. As of right now, the AI is primarily dependent on the introduction quiz presented to the user at the beginning along with a handful of parameters that are either explicitly set or automatically set based on usage [15]. How much these values should be changed along with how often and when they should remain to be seen. This challenge must also be reconciled with the need to maintain an efficient backend that can support the database operations and bandwidth costs necessary to scale with use. Other limitations are mainly feature-related details as the post-creation experience and the methods for students to communicate with organizations can be more streamlined and robust to meet the needs of the people using the application.

To solve the AI’s limitation on adaptation, more tools and processes will need to be put in place to better follow and tweak the amount and frequency of changes the AI should be made to best serve the needs of the user. Once we have a more complete picture, automation measures can be put in place accordingly.

REFERENCES

[1] Gupta, Shashank. "Sentiment analysis: Concept, analysis and applications." Toward Data Science (2018).
[2] Payne, Rap, and Rap Payne. "Using Firebase with Flutter." Beginning App Development with Flutter: Create Cross-Platform Mobile Apps (2019): 255-285.
[3] Nandwani, Pansy, and Rupali Verma. "A review on sentiment analysis and emotion detection from text." Social Network Analysis and Mining 11.1 (2021): 81.
[4] Gajarla, Vasavi, and Aditi Gupta. "Emotion detection and sentiment analysis of images." Georgia Institute of Technology (2015): 1-4.
[5] Shani, Guy, and AselaGunawardana. "Evaluating recommendation systems." Recommender systems handbook (2011): 257-297.
[6] Doshi, Abhishek. "How to publish your first Flutter package!!." Jaypee University of Engineering and Technology, Raghogarh, Guna (MP): 44.
[7] Christian, David, Amy Lawrence, and Nicole Dampman. "Increasing college access through the implementation of Naviance: An exploratory study." Journal of College Access 3.2 (2017): 28-44.
[8] Rosa, Renata Lopes, et al. "A knowledge-based recommendation system that includes sentiment analysis and deep learning." IEEE Transactions on Industrial Informatics 15.4 (2018): 2124-2135.

[9] Chauhan, Anmol, Deepank Nagar, and Prashant Chaudhary. "Movie recommender system using sentiment analysis." 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM). IEEE, 2021.
[10] Bhanuse, Roshan, and Sandip Mal. "A systematic review: deep learning based e-learning recommendation system." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021.
[11] Tusar, Md Taufiqul Haque Khan, and Md Touhidul Islam. "A comparative study of sentiment analysis using NLP and different machine learning techniques on US airline Twitter data." 2021 International Conference on Electronics, Communications and Information Technology (ICECIT). IEEE, 2021.
[12] Nasukawa, Tetsuya, and Jeonghee Yi. "Sentiment analysis: Capturing favorability using natural language processing." Proceedings of the 2nd international conference on Knowledge capture. 2003.
[13] Gan, Qiwei, and Yang Yu. "Restaurant Rating: Industrial Standard and Word-of-Mouth--A Text Mining and Multi-dimensional Sentiment Analysis." 2015 48th Hawaii International Conference on System Sciences. IEEE, 2015.
[14] Alsaeedi, Abdullah, and Mohammad Zubair Khan. "A study on sentiment analysis techniques of Twitter data." International Journal of Advanced Computer Science and Applications 10.2 (2019).
[15] Kurniasari, Lilis, and ArifSetyanto. "Sentiment analysis using recurrent neural network." Journal of Physics: Conference Series. Vol. 1471. No. 1. IOP Publishing, 2020.