Siena College's Institute of Artificial Intelligence
TREC 2015 Contextual Suggestion Track

Aidan Trees
Siena College
515 Loudon Road
Loudonville, NY 12211
an07tree@siena.edu

Tom Heritage
Hartwick College
PO Box 4020
Oneonta, NY 13820
heritaget@hartwick.edu

Kevin Danaher
Siena College
515 Loudon Road
Loudonville, NY 12211
ka29dana@siena.edu

Zach Siatkowski
Siena College
515 Loudon Road
Loudonville, NY 12211
z08siat@siena.edu

Darren Lim
Siena College
515 Loudon Road
Loudonville, NY 12211
dlim@siena.edu

Abstract

An overview of Siena College's participation in the Contextual Suggestion track of the Twenty-Fourth Text Retrieval Conference (TREC) is provided in this report. Our goal was to first design a search technique for complex information on specified POI's (points of interest) from a collection set given by TREC. The second part of our task was to return a list of ranked suggestions dependent on a given context and a user's interests. Multiple API's were utilized for information retrieval on each particular POI including Google Places, Foursquare, and Yellow Pages. This process was repeated for not only the POI's being suggested to the user, but for the POI's rated by each user as well. From this information, profile preferences were created for individual users by examining the categories of the POI's that they had rated. To build these preferences, we designed a scoring algorithm to associate a value with each individual category returned by the API's. We finally created a ranking system that includes a unique penalty function to sort our suggestions of attractions specific to each of the users' interests.

1. Introduction to Contextual Suggestion

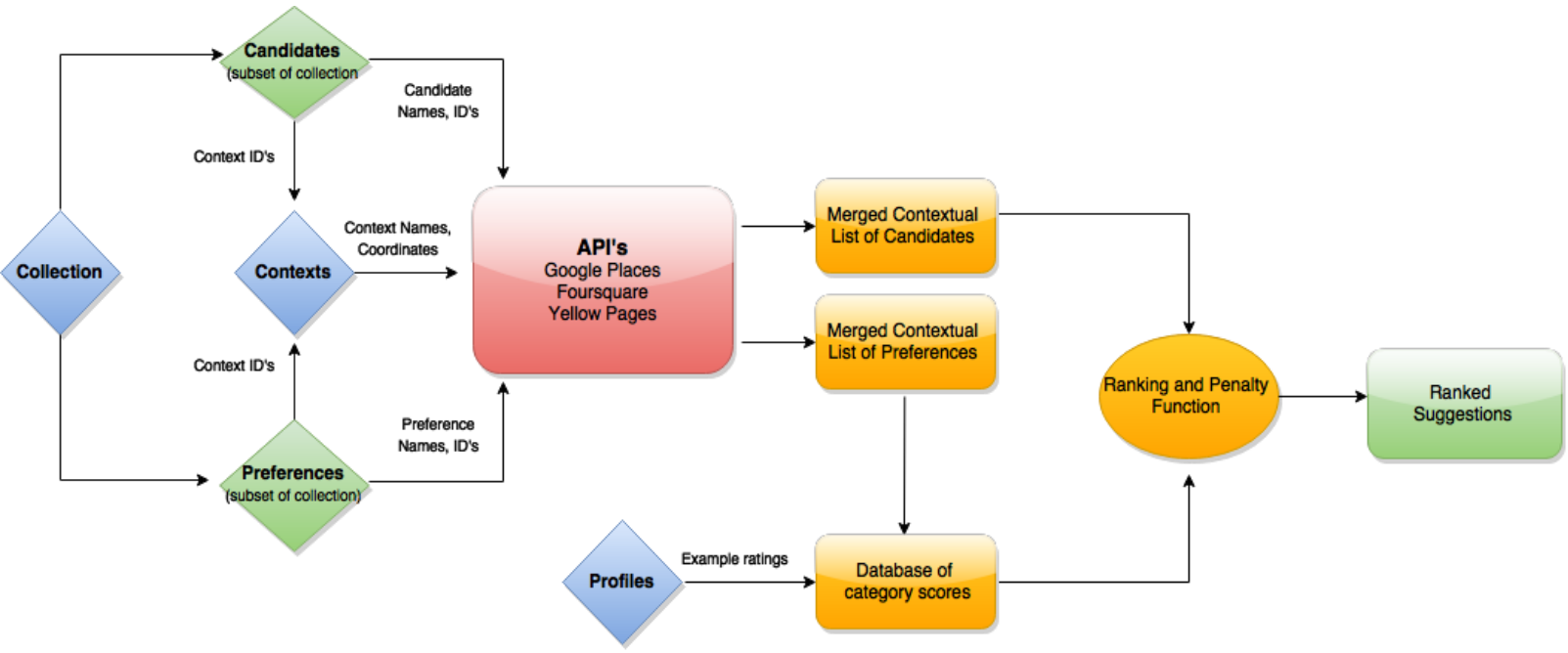
The Text Retrieval Conference (TREC) is a program that originated in 1992 and focuses on various forms of information retrieval, sponsored by the National Institute of Standards (NIST) and U.S. Department of Defense. TREC's purpose is to promote research in the information retrieval field by providing the infrastructure needed for big scale evaluation of information retrieval methodologies. Siena's Undergraduate Computational Contextual Evaluation and Suggestion System (SUCCESS) team participated in the TREC 2015 Contextual Suggestion track. This track involves an area of information retrieval that evaluates search techniques for complex information needs, which are strongly dependent on contexts and user interest. Such a system could suggest several places of interest in a location foreign to an individual based on his or her favorite hobbies and places to go. Our goal was to create a contextual suggestion system that returned a set of ranked suggestions of things to do in particular location for hundreds of individuals based on their profiles.

TREC initially gave our team 150 contexts(locations) in which a set of suggestions could possibly be asked for, each containing a context ID, city, state, latitude, and longitude. We were then given a collection of over one million attractions. These included the attractions that were rated by individuals, and also the attractions we would be suggesting to them. Each attraction consisted of an attraction ID, title, URL, and context ID corresponding to its location. Lastly, the profiles of individuals were released, each with a list of attractions that they've rated. Individuals could rate POI's from 0(strongly uninterested)-4(strongly interested) or -1 if they didn't leave a rating.

Absorbing all of the given information, our SUCCESS team at Siena College designed an approach that compared information from three different API's: Google Places, Foursquare, and Yellow Pages. After a significant amount of data manipulation, we were able to suggest a ranked list of attractions for each profile-context pair.

2. Approach

Our team's approach to the Contextual Suggestion track was to focus solely on categorizing user tastes and categorizing individual attractions. Our use of categories for user-rated POI's was a choice that we made both for simplicity but also because it accurately identifies what users like and dislike. Categories will also bring out the features of an attraction that will influence the interest of a user, and signify if the attraction should have a higher position in the ranked list of suggestions. In order to break down an attraction into several categories, we sent each one through our API system. We then merged the categories returned by the API's into one list. Based on rated POI's given by the profiles, we could associate a score to each category, creating a table of category values specific to each person. Using this table, we can rank suggested attractions by scoring their categories according to the customized category scoring table.



3. POI Retrieval System

The first API we evaluated for use in our contextual suggestion system was Yelp. However, after researching their terms of use and contacting the company, we discovered ranking their information would be contradicting their user agreement. We decided to disregard Yelp for this project. Instead, the three API's we used to categorize the attractions were Yellow Pages, Google Places, and Foursquare. Using the attraction name, city name, or location coordinates depending on the API, all three of them were able to return information on the majority of the attractions, including a list of categories. Though we were only interested in the categories they returned, other information was used to compare the results and determine if the attraction was indeed the same one from the TREC collection. This information included name, and latitude and longitude coordinates. If one of the three API's differed from the original in the TREC collection, those categories weren't taken into consideration. From all of the API's that returned a match with the original attraction, the categories were added to one combined list. Duplicates and plurals were disregarded, along with general categories such as, "establishment" or, "point of interest". At the end of API querying, each user-rated attraction and potentially suggested attraction had a list of categories to identify it.

4. Data Manipulation

Profiling & Attraction Scoring

For all profiles, a list of select attractions were provided that the person rated. To create their preferences, the rating was stored for each attraction and was used to give a numeric value to each of the categories associated with that attraction.

For example, if a user gave Madison Park a rating of 0, the "park" category and and all other associated categories with Madison Park would be given a score of -4. If the rating given to the next attraction was a 1, the category scores associated with that attraction would be deducted by -2. A rating of 2 increases category scores by +1, a rating of 3 increases category scores by +2, and a rating of 4 increases category scores by +4. As category scoring occurred, we kept track of the frequency that each specific category appeared. This was done in order to take the average score for each category and give those average scores to each profile. This way for each person we built a table of categories, each with one numeric value based on their attraction rankings. Now that we had category scores for each attraction, we had to give one value to each attraction in order to rank them.

We scored the attractions by adding up the total of the pre-calculated category scores from the individual attraction. We then divided this result by the number of categories that attraction had to get a score. The average was taken rather than the sum due to the fact that some attractions contained more categories than others. We ignored certain categories when scoring attractions that were too broad to be meaningful (such as "establishment").

Ranking Attractions & Penalty Function

After scoring the attractions based on average category scores, we sorted the entire list of attractions for a given profile. However, rather than suggesting them directly in descending order, we first implemented our penalty function. To emphasize variety in our

suggestions we penalized previously seen categories. For example, if the first result was a bar, the bar category's score was decremented for all of the following attractions. This allowed other attractions containing categories that were rated slightly below the "bar" category to move their way up the rankings. In order to conduct this approach, we first sorted the attractions based on score before the penalty function took place. We then went down the list in order from first to last, determining if a category in each attraction was seen in a previous attraction. We decremented the repeated categories' scores slightly each time they were seen and finally gave the attraction its new total value. We then sorted all of the attractions based on their scores one final time.

The reasoning to promoting variety is that it is a more natural suggestion methodology and it also provides a safety net to prevent suggesting the same kinds of attractions to a user if he gets tired of them. We run the risk that the highest rated category is not the best category for that profile. With a penalty function, not one single category could dominate the complete list of suggestions.

When a person suggests numerous places to go their friend, he'll suggest different types of attractions. The person suggesting places may be unsure of what his friend enjoys, so he instinctively increases the chance of suggesting relevant attractions by implementing variety. The utility of suggesting a bar decreases once one bar is already suggested. It can be assumed that when activities are being suggested, only a subset of them can be done. If the person provides his friend with the best bar to go to, the chance that his friend would go to the second best bar slightly decreases due to the similarity between the attractions. If the option of a different type of activity was provided, such as going to a movie theater, the utility of second suggestion becomes higher.

5. Results and Conclusion

	Run A	Run B
P5	**0.5403**	**0.5564**
P10	**0.4986**	**0.5204**
P15	**0.4720**	**0.5058**
P20	**0.4604**	**0.4775**
P30	**0.4215**	**0.4215**
P100	**0.1264**	**0.1264**
P200	**0.0632**	**0.0632**
P500	**0.0253**	**0.0253**
P1000	**0.0126**	**0.0126**
MRR	**0.6983**	**0.6995**

We submitted two runs: SCIAI_runA and SCIAI_runB. Our average results from both of these runs are shown in the table below. Run A utilizes our extra penalty function as described in section 4b. Our goal with the two separate runs was to see how much adding in the penalty function affected the accuracy of our results. As you can see in the table, our results were very similar in both runs, with Run B being slightly more accurate. Run B excelled over Run A in precision at 5, 10, 15, and 20, as well as in the Mean Reciprocal Rank(MRR). Surprisingly the results of precision at 30, 100, 200, 500, and 100 are identical between both runs. This is most likely due to POIs with low ranked categories always being put farther down the list. Our total precision at 5 from both runs is right in between the average and best total precision at 5. Our

MRR from both runs was slightly above the average. In conclusion, both of our runs gave us above average results and we learned that our program is more accurate without the penalty function.