Restaurant Search with Predictive Multispace Queries

Alexei YATSKOV
Keio University
Graduate School of Media and Governance
5322 Endo, Fujisawa
Kanagawa, Japan, 252-8520
alex@yatskov.com

Yasushi KIYOKI
Keio University
Graduate School of Media and Governance
5322 Endo, Fujisawa
Kanagawa, Japan, 252-8520
kiyoki@sfc.keio.ac.jp

ABSTRACT

This paper describes a web-based search application used for locating restaurants in a multidimensional content space via interactive space visualization. The primary goal of our research is to reduce the cognitive complexity of query selection, as well as to help the user avoid one of the common pitfalls of traditional search mechanisms: retrieving too many or too few results. Our method approaches this problem by continuously staying one step ahead of the user, constructing a graphical output to summarize how further query adjustments will impact subsequent search results. In actively precomputing queries ahead of the user we help them avoid the trial-and-error search process often associated with trying to find something in an unfamiliar, opaque database. We combine our visual search method with a profile-based knowledge system which helps users find restaurants accessed by people with similar interests.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: User Interfaces

Keywords

semantic search, restaurant search, visualization, forecasting, multidimensional

1. INTRODUCTION

Finding relevant results in a sea of information that we have come to refer to as big data is an ongoing challenge for users. When performing any kind of search, it is frequently the case that the number of results retrieved falls at one of the polar ends of the spectrum — none or simply too many to count, resulting in information overload[5]. Frustrated users, lacking context regarding the content and distribution of data, are often left spending their valuable time attempting to locate desirable results by trial and error. As an increasing number of businesses and services make their databases available and searchable online, the ability to obtain sought-after results becomes all the more critical.

Our research views these difficulties as being a direct consequence of the user's lack of familiarity with the search space. While many types of search operations can be readily accomplished if one has specific, pre-existing knowledge of the dataset they are querying, it is not feasible to expect this level of knowledge from everyone. For this reason, we see the ability to preemptively familiarize users with an abstract overview of the database contents to be an important feature of a search system. In order to make the best use of the user's attention and to minimize the quantity of information they would have to process, we focused on building a highly visual, interactive, and intuitive search space visualization system. In addition to being able to visually designate the search query and inspect the graphical search output, we felt that it was important to provide a mechanism for guiding users towards results which would be of possible interest to them. When performing search queries in the real world, people unknowingly obtain contextual *hints* from the behavior of individuals around them[2]. This can be exemplified by people lining up to buy a product or crowding around a tasty restaurant. This feedback is not available online — users performing similar actions are never made aware of each other's presence, and cannot leverage each other's knowledge in decision making. In order to bring this property of the physical world into cyberspace, we designed a mechanism which allows the user to make decisions which leverage the *group knowledge* of similar individuals.

In order to demonstrate the advantages of our approach, we constructed a new application which allows its users to locate restaurants in a semantic space. This field was chosen for our research as forming a restaurant query is a task that often requires the user to make decisions about what criteria are important to him or her, often making compromises in the process (how far do I want to walk and pay for a tasty meal?). Our test data consists of over a thousand restaurant reviews from Yokohama, Japan, and introduces contextual properties such as accessibility and distance in addition to traditional ratings based on categories like value, taste, etc. We used this data set to construct a multi-dimensional orthogonal vector space which we allow to the user to navigate using a purpose-built web-based graphical interface. When executing a search query, the user transmits six key parameters, alongside profile information, which is then used to compute the search output. The search engine, in addition to outputting the results corresponding to the user query,

iiWAS '15, December 11-13, 2015, Brussels, Belgium

DOI: http://dx.doi.org/10.1145/2837185.2837193

pre-computes numerous variations on the inputted parameters in order to present a *forecast visualization* that the user may leverage when making adjustments to their query.

Effectively visualizing data within spaces of high dimensionality is an area that presents many challenges to computer scientists[1]. Once we cross over the threshold of the familiar three dimensions, we are forced to employ increasingly abstract forms (colors, shapes, etc.) to convey the concept of additional dimensionality. While these methods indeed make it possible to output a graphical representation of a fixed number of dimensions, the resulting visualizations may be difficult to comprehend to the untrained user. The problem becomes even more complex when we must visualize arbitrary dimensionality in an easy to understand manner. Our research solves these problems by presenting to the user a limited *dimensional profile* in favor of full space visualization.

In this paper we will introduce the restaurant search system prototype that we have built to demonstrate our technology, illustrating the distinct scenarios in which *forecast visualization* and *dimensional profiling* enable users to construct queries which lead to highly desirable results. We will discuss the performance characteristics and limitations of our approach, presenting ideas we have on possible future improvements to the system.

2. RESEARCH BACKGROUND

Computer science research into search systems traditionally has the tendency to focus on improving the performance of server software, with less interest shown towards improving ways of expressing results to the user; perhaps this is why search engines for services ranging from online shopping to restaurant selection are still primarily text-based. While there has been much interesting research conducted[7, 6, 3] into building multi-dimensional content representation schemes and new visualization systems, there has not been significant mainstream adoption. This may be in part due to the fact that building effective visualization systems to represent multidimensional content spaces is a challenging task with many usability considerations[9]; after all, the human perception system was designed for interacting with physical objects in the real world and not for grasping abstract objects and concepts[1].

We see the field of restaurant search to be an area where it is very natural to want to store restaurant information in a multidimensional space, while exposing an interactive query interface which enables new content discovery for our system's users. As prior research shows that there is value in developing cooperative features[2], we wanted to investigate the idea of multidimensional search for not only locating restaurants directly but also seeking out users with similar interests and leveraging their knowledge to improve the efficiency of the query creation process.

Our method's design was influenced by the work done on the *Mathematical Model of Meaning*[4]. The original research consisted of an image search system which used keywords to position content in a multidimensional space. The keywords in turn, were described as vectors composed of core *features*, a set of descriptors making up the axes of the content space. The idea of forming queries based on intent as opposed to forcing the user to input specific parameters was attractive as a direction for our research; restaurant search is an example of an activity where users must make frequent trade-offs and compromises based on numerous physical and informational factors, the combinations of which cannot be easily be described by words alone.

3. SYSTEM ORGANIZATION

Our system was developed as a server-client architecture, with the web front-end built as an HTML5 web application which utilizes SVG for visualizing results; the back-end was written Go, using MySQL as the underlying data storage mechanism. The client software can be used from nearly all operating system for which a modern web browser is available; GPS-enabled devices are able to take advantage of the geo-physcal search functionality. The server maintains a store of *restaurant data* which web clients can query via AJAX. Clients keep a persistent copy of user *profile data* in the browser local storage. The system does not employ a user database on the server of any kind, and as such registration is not needed.

3.1 Restaurant Data

Our research focuses on building a predictive search system which enables the user to locate desirable restaurants represented in a content space via a series of parameters. The data items stored in the system are described by six key vectors (from here on out referred to as *features*) each representing a specific meaning about a restaurant. Feature vectors are normalized to values between -1 and +1 and make up the axes of the six-dimensional restaurant search space. Positive feature values correspond to varying degrees of agreement with the associated meaning and negative values indicate disagreement; null values represent irrelevance.

The key features and the meanings they represent are as follows:

accessible
: ease of access via public railway

accommodating
: level of customer service and flexibility

affordable
: reasonableness of the overall price point

atmospheric
: the appearance and mood of the environment

delicious
: the overall quality of the food served

nearby
: the proximity of the restaurant to the user

3.2 Data Acquisition

In order to obtain the base review data for the features corresponding to `accommodating`, `affordable`, `atmospheric`, and `delicious`, we developed a custom data mining application to harvest reviews from a travel-related website, *TripAdvisor*. *TripAdvisor* is a traditional text-based search engine which maintains a database of user generated travel-related content and contains collections of restaurant reviews organized by region; we focused our scope on the businesses located in the city of Yokohama, Japan.

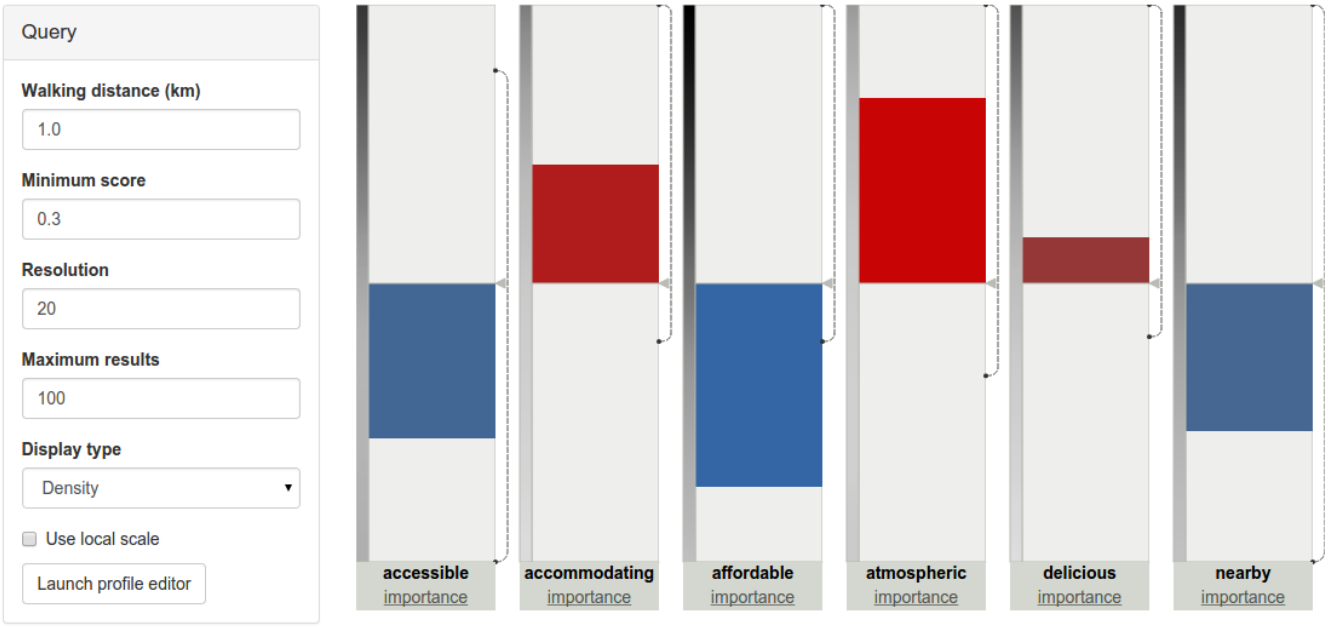

Figure 1: The user executes queries and views search forecast information through an interactive web application. They are presented with sliders which they can use to mutate the values of the six key *features*, the components of which are individually described in *Figure 3*. The left-hand side side of the window contains controls which users can interact with to operate the search and visualization systems.

3.3 Data Normalization

The raw data we obtained from *TripAdvisor*, after being stripped of unusable data (closed restaurants, entries without ratings) had to be normalized to work with our system. The data obtained was based on a rating system of 0 to 5 stars; each category was normalized to a -1 to +1 scale, thus making it consistent with the other features in the database. Every restaurant entry contained a street address, and as part of the conversion process, a latitude/longitude pair was computed for every location via the *Google Geocoding API*.

3.4 Data Precomputation

Computing accessibility for each restaurant location required us to obtain the distance from each restaurant to the nearest railway station (for the purposes of our research we assume that the user will be traveling on JR trains). We used a publicly available list of station names in Japan to locate and compute latitude/longitude pairs for all stations. We then applied the *Haversine Formula*[10] to compute great-circle distances between each restaurant and the closest station. It should be pointed out that this distance is only an estimate and does not take into consideration natural and man-made obstacles which can make the actual travel distances be greater. Actual route calculation (such as the turn-by-turn navigation provided by Google Maps), while undoubtedly providing accurate distance estimates, would lead to a significant increase in computational load on the server and as such was not utilized in our application.

4. USER INTERACTION

The user interacts with the search system through a dynamic web interface. He or she is presented with a screen that displays to them the six key features, represented as vertical sliders, shown in *Figure 1*. The sliders are initially centered on a starting value of zero but have a range of motion spanning -1 to +1, The user can interact with the sliders by clicking on them, thereby specifying the value of the corresponding feature. All of the features, when combined correspond to a point in six-dimensional space which is used as a starting point for locating nearby results in semantic space. Every time a feature value is changed, a query is sent to the server and executed. Results, along with predictive information for future queries are returned to the client which displays them to the user. A control panel allowing the setting of additional search parameters is provided along the left side of the interface.

4.1 User Profile Data

Users of our system can use the built-in profile editor to input relevant information about themselves which is used to build a *compatibility profile*. The editor interface presents the user with a series of questions which he or she can then reply to from the multiple-choice answers “Agree”, “Disagree” or “Neither” using a modal dialog as shown in *Figure 2*. The questions presented are the same for all users, with new entries being addable by anyone at any time. For example, someone who drives a car might add the question “I require parking”; someone who enjoys spicy food might write “I like spicy food”. All of the answers start out initialized to the default value of “Neither”, and thus do not have any influence in compatibility profile calculation. Users can answer as many or as few questions as they desire, depending on relevance to their search query. By answering these questions, users are grouped into similar categories, which in turn are used as basis for helping them discover what kind of restaurants people similar to them enjoy[8].

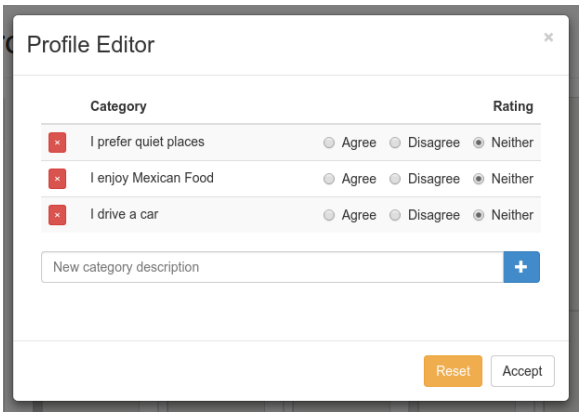

Figure 2: The user interface of the profile editor displayed in a dialog; users can answer as many or as few of the questions presented to help them to find restaurants accessed by similar individuals. New questions can be contributed by anyone and unreferenced question data can be freely deleted by users.

4.2 Forecast Visualization

One of the key features of our technology, *forecast visualization* allows the user to see how changes to their search query will impact the results *before* they take any action. While visualizing the contents of a multidimensional space past the first three dimensions is difficult to do in a clear, easy to understand way, our system accomplishes this by displaying a *dimensional profile*. The dimensional profile can be described as a list of samples taken at set intervals across the range of the feature, with information about the query results at that each step recorded in the process. We use linear interpolation to create an SVG gradient which serves to visually forecast the changes in results brought about by user modification of a feature value. The color of the gradient ranges from white to black, corresponding to low and high values respectively, thus representing varying levels of *data pressure*. For our research, we apply *forecast visualization* to two different result attributes `density` and `compatibility`; the user can alternate between these visualization modes by selecting the corresponding option in a drop-down list. The major components of the feature visualization interface are identified in *Figure 3*.

4.2.1 Density Visualization

The density visualization mode allows the user to observe variation in the number of results returned in response to changes to feature values. This functionality allows the user to fine-tune the query to get a desirable number of matches without having to resort to trial-and-error. If the returned set of restaurants is unmanageably large, the user can tighten his or her search criteria to better correspond to their ideal restaurant by increasing the minimum score threshold. On the other hand, if there are not enough options to select from, the user can easily decide on a strategy for loosening their requirements to get more matches (either reducing the minimum score threshold or moving feature sliders to areas with higher data pressure). Other benefits of this visualization technology include the ability to see trends and correlations in the data. In the case of the data

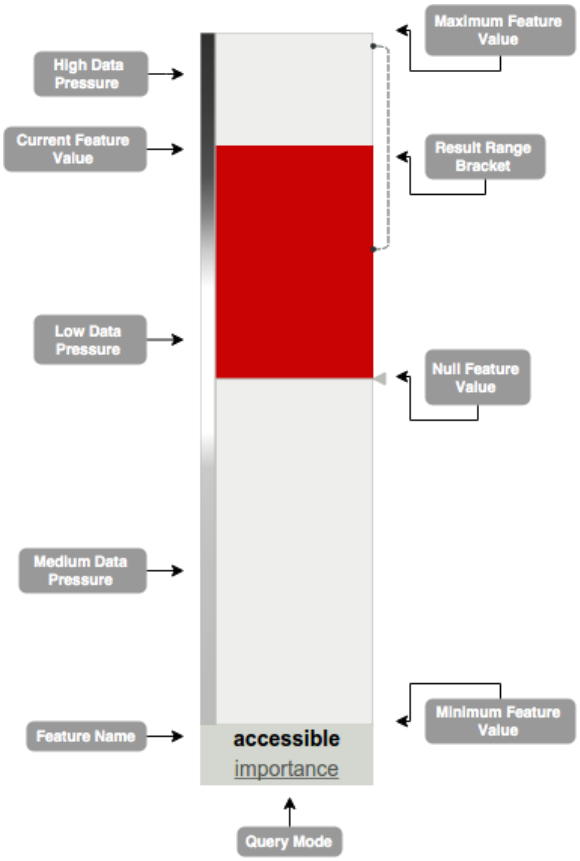

Figure 3: Illustration of the major components of the widget used for feature value selection and forecast visualization. The user can pick a value for the current feature by clicking anywhere on the slider, which represents a possible range of +1 (at the top) to -1 (at the bottom), with 0 in the middle. The current search mode can be toggled between `importance` and `similarity` by clicking the corresponding underlined text.

set that we are using from *TripAdvisor*, for example, we noticed high review densities at the upper and lower end of feature value ranges, corresponding to highly polarized reviews. We assume that this may be attributed to the fact that the majority people do not take the time to write about their experience at a restaurant unless it was a very good or bad one. Being able to see inherent biases in the reviews was one of the things not possible to do with the original *TripAdvisor* text-only search interface to this data.

4.2.2 Compatibility Visualization

When operating in this mode, the forecast visualizer displays the compatibility density for each feature. This function enables the user to see the estimated quality of results, calculated based on how many users with a similar profile have accessed the reviews in question. For example, the user might realize that by changing the value of the "nearby" parameter to expand their search to restaurants further away from their position, they may be able to find a location that is highly popular amongst people with similar profiles. The advantage of this method of forecasting is that users who may not have a clear picture of what kind of restaurant they are looking for can leverage the system's group knowledge to find suitable locations, while at the same time maintaining control of the search parameters used for the query.

4.3 Result Output

Results are output in a tabular format, with the following columns: `name`, `distance to user`, `closest station`, `distance to station`, `compatibility`, `score`, and `access count`. The user is able to sort the output in ascending or descending order by clicking on a column; clicking on the restaurant name allows the user to learn additional information about the business by visiting its *TripAdvisor* page. They can see how many results are currently displayed and compare this number to the quantity that was matched by their query. When the user opts to visit the review page corresponding to one of the search results, it is opened in a new browser window. The still-running search application executes an AJAX request to the search server, transmitting the user's profile information to be stored as an access request for purposes of future compatibility calculation.

4.4 Customization

Our system features several options that the users can adjust in order to modify the search query behavior and visualizations. Unless otherwise noted, changing any of these settings will immediately trigger the execution of a new query and cause new results to be displayed. The behavior of these customizations is described from the user's perspective; a technical explanation will be provided later in the *Search Mechanism* section.

4.4.1 Walking Distance

In processing the value of the `accessibility` feature, our system needs to know the maximum distance that the person executing the query is willing to walk from a train station to a restaurant. This parameter allows the user to specify a preferred distance in kilometers; a value of 1.0 kilometer is provided as the default.

4.4.2 Minimum Score

When performing their search, the user can specify the minimum score threshold that results must obtain to be displayed. Higher scores translate to higher correlation to the user's query, in exchange for a smaller number of results displayed. This parameter can be freely set to any value, but in our experiments, values in the range of 0.0 to 2.0 appeared to be optimal. The parameter is initially set to a value of 0.25, allowing for slightly loose queries by default.

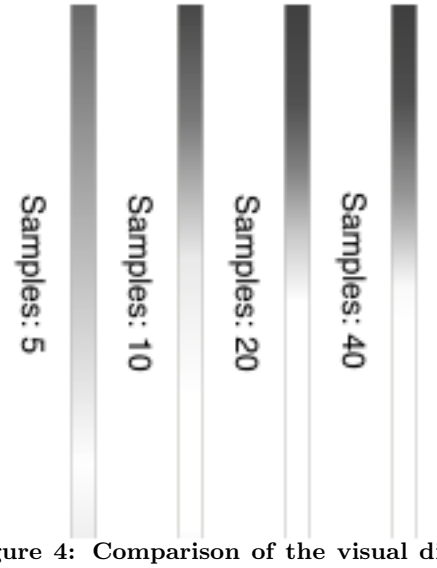

Figure 4: Comparison of the visual differences between using four different resolution settings to represent an identical content space. Our system utilizes linear color interpolation to fill in the gaps between samples; lack of data leads to more interpolation, visualized as smoother gradients. Higher resolution allows the system to build a more sharply defined picture of the content space at the expense of additional computation.

4.4.3 Resolution

This parameter determines the sample frequency used in building the *forecast visualization* data for each feature. Lower resolutions translate to faster queries but higher values improve accuracy. The default setting provides twenty samples for every feature, which in our usage was found to be adequate for most searches. A graphical comparison of forecast visualizations rendered at 5, 10, 20, and 40 samples is illustrated in *Figure 4*.

4.4.4 Maximum Results

In order to avoid an unmanageably large number results being displayed in the client in the case of a overly loose query, our system allows the user to set a cap on the maximum number of results returned. The result set is sorted before being trimmed on the server, ensuring that only the results with the lowest scores are hidden from the client.

4.4.5 Display Type

This toggle allows the user to switch between the density and compatibility visualization modes. As the forecast data required for displaying both visualizations is computed whenever a query is executed, changing this setting does not require any additional server processing. By default, we

display the result density to the user.

4.4.6 Local vs. Global Scale

Our system can generate the forecast visualization in either `local` or `global` scale mode. When in local mode, we use the local minimum and maximum values to represent white and black on the gradient, respectively. In global mode, the minimum and maximum values across all of the features are used for gradient color calculation. Local scale mode is useful for being able see changes that occur from modifying a given feature value, no matter how small their contribution is to the overall query result. The global scale mode, on the other hand, is more meaningful for users deciding what features have the greatest influence on the search results. This property is particularly helpful when we are faced with too few or too many results returned by a query — we can quickly find and modify the "bottleneck" feature that is leading to an undesirable number of restaurants being matched. As the gradients created when using the global scale correspond better to the number of returned results, it is the default setting.

4.4.7 Importance vs. Similarity

Every feature slider contains a toggle that specifies how the feature should be interpreted by the search system. The `importance` mode conveys the value of a feature's meaning to the user. For example, positive values for the `delicious` feature imply varying degrees of importance in matching restaurants with high quality food; in contrast, negative values correspond to intent to find locations with food that has been rated to be poor. The `similarity` mode operates differently in that instead of describing how much *emphasis* a feature should have in calculating a restaurant score, it provides a specific value which should be searched for. This difference is easiest to illustrate when examining how the two modes treat the feature value of zero. In `importance` mode, a value of zero implies that the given feature is not important to the user and as a result, it does not factor in during the computation of a restaurant's score. When in `similarity` mode, selecting zero would imply the user's intent of finding restaurants with an average rating of zero for that particular feature. As it can be more intuitive to think about features as being relative to each other rather than absolute values, all of the features are set to the importance mode by default.

4.4.8 Result Brackets

When restaurant results are displayed to the user, we render brackets to express the range of values matched for every single feature within three standard deviations. Doing so allows the user to visually see the distribution of the result set, and provides awareness to how well each feature was matched in the query, allowing him or her to judge the overall quality of the search. Furthermore, the bracket size is a good indicator of the "tightness" of the search; loose bounds usually imply that we can get better results by increasing the minimum score value.

5. SEARCH MECHANISM

When the user directs the web client to perform a search, an AJAX request is sent to the server which then processes the query. The request consists of not only the feature values and search preferences, but the user's physical position and profile data as well. Positional information is obtained through the `geolocation` API available in modern browsers. If this information is not available (browser incompatibility or the user refuses to provide it), the query can still be executed without it (albeit the `nearby` feature cannot be calculated). The query is processed in a series of steps, leading to the calculation of a score for each restaurant. Restaurants with scores higher than the user-assigned minimum threshold are returned to the clients as results.

5.1 Dynamic Feature Computation

While most of the features used in queries are stored as static data in the database, the `nearby` and `accessible` features depend on search options and the user's geographic position and thus must be computed whenever the user performs a search. The dynamically computed features are treated identically to the static ones stored in the database during the restaurant score calculation step.

5.1.1 Calculating the `nearby` feature

Given that we have the latitude and longitude data provided to us in the search query, it is possible to compute the Haversine Distance[10] between the user and the restaurant locations stored in the database. As the resulting distance is a measure of length, we must normalize this value in order to be able to handle it as a feature within our search system. The normalization function for any given restaurant can be expressed as

$$nearby(d_{user}) = -\left(\frac{d_{user} - d_{close}}{d_{far} - d_{close}} - 0.5\right) * 2$$

where d_{user} is the distance to the user, d_{close} is the distance to the closest restaurant, and d_{far} is the distance to the farthest restaurant from the user. This function effectively treats the distance to the closest restaurant as the definition of "nearby", and considers the restaurant farthest away to be the opposite of this feature's meaning.

5.1.2 Calculating the `accessible` feature

For our purposes, the distance from the nearest station to a restaurant is the factor used for determining the value of the `accessible` feature. When executing the search query, the user specifies the maximum distance that they are willing to walk; we use a normalization function to obtain a value for this feature from distance for a given station expressed as

$$accessible(d_{station}) = \max\left(-1, \min\left(1, 1 - \frac{d_{station}}{d_{walk}}\right)\right)$$

where $d_{station}$ is the distance to the closest station and d_{walk} is the maximum distance that the user is comfortable walking to their destination. When the distance to the station is the maximum distance, the above function equates to zero; it is an edge case where the station is neither particularly accessible nor inaccessible. Shorter distances compute to positive accessibility values and longer distances to negative ones.

5.2 Calculating Compatibility

Features are not the only data which must be processed for every query; profile data sent by the client must be processed and compatibility scores must be obtained for the restaurants being searched. This step requires several queries to

be executed and is the most performance intensive step in the search process.

When requesting a search query, the client transmits the user's profile data, represented as a series of key-value pairs corresponding to the question and answer data that they optionally completed before starting search. The profile keys correspond to question identifiers, represented by a unique number; the values are answers represented by the set consisting of -1, 0, or +1, corresponding to the replies "Disagree", "Neither", and "Agree" respectively. For each restaurant that is processed in our query, we read the access history, and compute a compatibility score based on the difference between the current user profile and the ones stored from previous searches. This is accomplished with the function illustrated in the pseudocode below:

```
func compatibility(features1, features2) {
    result = 0

    for name, value1 in features1 {
        if name in features2 {
            result += value1 * features2[name]
        }
    }

    return result
}
```

From the above code we can see that unanswered questions, defaulting to the baseline "Neither" answer do not have any influence on the compatibility score. The value computed for compatibility is in fact only changed when the answer from the user's profile agrees or disagrees with the one in the restaurant's access history. This approach to score calculation, combined with that all the answers to the profile questions are initialized to "Neither" permits the user to answer only the questions that they see as relevant to their search experience, without their results getting skewed every time a new profile question is registered in the system. Upon performing this computation between the user's profile and all of the items in the access history, we can use the average score as the final compatibility value for the restaurant.

5.3 Score Calculation

To determine whether or not a given restaurant should be returned as a result to the user, our system computes a score which is then compared to the minimum threshold setting. The calculation is similar to the compatibility computation in that we compare two feature sets to determine the final score. As described earlier, the features consist of static and dynamic data, numerically representing a restaurant's properties such as `delicious` and `accessible`.

Each feature's contribution to the restaurant's score depends on it's value and the mode setting selected by the user. As mentioned earlier, `similarity` is most helpful in finding restaurants closely matching the exact parameters specified by the user; `importance` is used to designate the significance of each feature during score computation. The behavior of the scoring function follows the pseudocode which is shown below:

```
func score(features1, features2, modes) {
    result = 0

    for name, value1 in features1 {
        value2 = 0
        if name in features2 {
            value2 = features2[name]
        }

        if modes[name] == importance {
            result += value1 * value2
        }
        elseif modes[name] == similarity {
            result += 1 - abs(value1 - value2)
        }
    }

    return result
}
```

5.4 Forecasting

Our system computes forecast data by sampling the possible -1 to +1 value range of each feature at intervals of $2/resolution$, while keeping the values of the other features constant. The number of overall samples taken can therefore be described as $features * resolution$. This method allows us to generate predictive information which can be used to build a visualization illustrating how modifying the value of any given feature will impact the search results. This approach is valid for search spaces of arbitrary dimensionality but for the sake of simplicity we present a visual example of forecast computation for a two-dimensional space in *Figure 5*. The black areas correspond to matched content and *red* and *blue* colors represent two distinct features, or dimensions. When calculating the forecast data for the *blue* feature we can observe how changing its value impacts the search results while keeping the value for the *red* feature constant. Computing data for the *red* feature works nearly identically with the exception that it is now the *blue* feature remains constant while we mutate the *red* feature. This can be more intuitively described as *shadowing* content wherever it is detected in a sample. This one-dimensional slice can include numerical information describing intersected content; for our system we focused on recording the number of restaurants at each sample (`density`) and average `compatibility`.

6. PERFORMANCE

The work performed by our system can be divided into two categories: *searching* and *forecasting*. Both operations can be readily optimized with parallelized execution; we heavily utilize *goroutines*, a technology similar to green threads to improve query throughput. The overall processing time varies from system to system, but in general searches complete in under 100 milliseconds when executed on the mid-range consumer hardware used for testing. We believe that temporary, per-user profile and location caching on the server could be a good way to attain significant performance improvements, but we have not yet conducted such tests as of the writing of this paper.

6.1 Searching

Our system's query execution time grows linearly with the addition of new restaurants to the data store. The search step consists of retrieving restaurant and access data from the database, and computing compatibility and accessibility for every record. While the actual calculations do not take

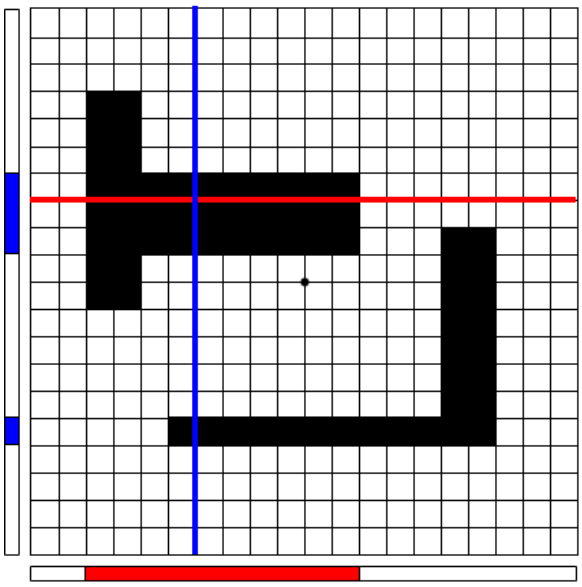

Figure 5: Visual representation of *feature shadowing* in a two dimensional content space. Black blocks represent content matches and red and blue lines correspond to samples taken at regular intervals in order to build prediction data describing how feature value changes modify the overall search result.

any significant amount of time in practice, the time required for repeated database operations add up to make this the slowest step in query execution.

6.2 Forecasting

Search forecasting, while requiring significant arithmetic computation, is easily parallelizable, with the execution time growing at a linear rate with the introduction of additional features. The `resolution` parameter specified by the user has a direct impact on the time required to complete a query and should be restricted to a reasonable upper bound to conserve server system resources.

7. OTHER APPLICATIONS

The presented technology, while for the purposes of our research centered on the area of restaurant search, can be readily adapted to work in other fields. Our method of using interactive visualization to proactively help the user reduce the amount of time lost to trial-and-error in query selection can be incorporated into other systems which perform search in a multidimensional space. Possible use cases include finding and displaying potentially dangerous chemical combinations in areas ranging from environmental study to pharmaceutical science. Other scenarios include the interpretation of trends and interactions in semantic computing; the text-only output of today's systems is particularly limited in situations where the developer must convey to the user the relevance of data expressed in "fuzzy" search results. Our technology does not impose arbitrary restrictions on the dimensionality of data and, as we have shown, can make use of dynamic parameters as query input, making its use possible across a range of applications.

8. CONCLUSION

In this paper we introduced a forecast-based search method and explored how it can help users discover restaurants in an unfamiliar multi-dimensional content space. We discussed how our interactive visualization technology can help users improve their query, narrowing down the number of returned results to a set best corresponding to their search criteria. To further mitigate the difficulties associated with searching an unknown dataset, we introduced the concept of using profile data to passively build group knowledge, thereby aiding users in finding restaurants accessed by people with similar interests to that of their own. To relate the applicability of our system to the real world we demonstrated how data from a traditional restaurant review website, *TripAdvisor* could be systematically harvested, supplemented with geophysical parameters, and converted for use within our semantic database.

Our technology's primary limitation is that it must be used for data sets which can be readily represented in a multi-dimensional space. We cannot, for example, meaningfully support search by restaurant name, which by itself has no inherent dimensionality on its own and thus cannot be used in our calculations. Another area for future improvement is the basis for recording the user's interest in a restaurant for the purpose of compatibility calculation. While accessing a review site in of itself shows a desire to learn more about the location, upon obtaining more information, the user may lose interest. We can mitigate this problem by displaying restaurant information in-line with results, requiring the user to press a button to obtain a critical piece of information, such as the street address. By doing so, we can obtain a higher degree of confidence that the result is indeed of practical interest to the accessing user.

If existing search engines were to transition away from the traditional, largely text-based search interface and adopt a proactive, visual approach to query construction such as the one described in this paper, it would be an important step in making the Web more discoverable and transparent to the user. We hope to see more systems taking advantage of modern web technology to take upon themselves the user's burden of trial-and-error search, leading the way to a better query building experience across many different fields.

9. REFERENCES

[1] G. Grinstein, S. Laskowski, and A. Inselberg. Key problems and thorny issues in multidimensional visualization. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 505–506, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[2] J. K. Hall and Y. Kiyoki. Identifying and propagating contextually appropriate deep-topics amongst collaborating web-users. In *EJC*, pages 146–157, 2013.

[3] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of the 6th Conference on Visualization '95*, VIS '95, pages 279–, Washington, DC, USA, 1995. IEEE Computer Society.

[4] Y. Kiyoki, T. Kitagawa, and T. Hayama. A metadatabase system for semantic image search by a mathematical model of meaning. *SIGMOD Rec.*, 23(4):34–41, Dec. 1994.

[5] C. López and R. Farzan. Exploring the mechanisms behind the assessment of usefulness of restaurant reviews. In *Proceedings of the 7th International Conference on Communities and Technologies*, C&T '15, pages 19–27, New York, NY, USA, 2015. ACM.

[6] J. Mothe and T. Dkaki. Interactive multidimensional document visualization. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 363–364, New York, NY, USA, 1998. ACM.

[7] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional databases. *Commun. ACM*, 51(11):75–84, Nov. 2008.

[8] V. Suresh, S. Roohi, and M. Eirinaki. Aspect-based opinion mining and recommendationsystem for restaurant reviews. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 361–362, New York, NY, USA, 2014. ACM.

[9] E. R. A. Valiati, M. S. Pimenta, and C. M. D. S. Freitas. A taxonomy of tasks for guiding the evaluation of multidimensional visualizations. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, pages 1–6, New York, NY, USA, 2006. ACM.

[10] J. Šeděnka and P. Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 99–110, New York, NY, USA, 2014. ACM.