Abstract

Researchers in the transportation community are interested in analyzing the trends of research in their field. A website where people from the community could post their research needs, monitor and vote on others’ research ideas, view visualizations of research trends in the past, could be helpful to make this possible. In this paper, we describe our website project, titled Transportation Research Analysis using Natural Language Processing (NLP) Techniques, or TRANS. We explain the methodology we used to obtain data from the Transportation Research Board (TRB), extract information from data using NLP techniques, visualize the extracted information and publish a website that provides the required functionalities.

1 Introduction

People in the transportation research community are interested in understanding the trends and dynamics of their research field, collaborating with colleagues over research ideas in a convenient setting, and posting their research proposals to the community.

Researchers also have concerns about the ineffective use of research funds, and believe that repetitive research in the field is causing a waste of time and money. Many research ideas and projects are repeatedly published with slight repackaging. Another need in this community is the categorization of research projects, which is useful to track the amount of research done in each sub-field, understand research trends within the community, and as a result bring researchers with similar interests together.

At the moment, these tasks are partially managed by the Transportation Research Board (TRB), however this is both costly and ineffective for specific tasks. Performing the tasks computationally will produce fast, cheap, and high-quality results. Visualizing these results will make interpretation and analysis easier, and will communicate them to a larger portion of the community.

In this paper, we describe our analysis and visualization website project, titled TRANS. The goal of TRANS is to provide researchers with the tools they need to analyze research dynamics in their field, monitor others’ research needs statements, and collaborate by voting on papers and research needs statements. Natural Language Processing (NLP) techniques are used to automatically categorize research needs statements and papers, discover repeated work, generate similarity lists in the website and therefore help people with similar interests get together. Information visualization methods are used to visualize the information discovered, and all of this is published on the TRANS website.

This paper is organized as follows: Section 2 lists related work, and Section 3 describes the data we use, how it is acquired and preprocessed. In Section 4, we give a detailed explanation of the methods used, and Section 5 describes our visualization approaches with screenshots, and demonstrates the final product: TRANS website. Some ideas for future work are included in Section 6, and finally Section 7 provides final remarks and concludes the paper.

2 Related work

There have been some work on analyzing citation networks, to discover information including the key role players in the field, interactions among groups, and how research trends emerge and change over years. Two projects by a group of researchers in the University of Maryland have focused on analyzing these issues.

The PopIT project is inspired by the assumption that, in any research area there are some concepts that are the main focus of researchers. The set of concepts that dominate the field change over time, along with the evolution of the field.
This project aims to find how the actions and opinions of the individual researchers are related to the generation and change of core concepts in the field. The name comes from the fact that they focus on Information Technology as an example field, although they claim that the idea is applicable to any other field ([19, 21, 24, 19, 23]).

Science & Technology Innovation Concept Knowledge-base (STICK) is a new project that aims to provide a database with visual capabilities to monitor and understand the innovations emerging in a research field. Three fields are used as examples in their work: information technology, biotechnology, and nanotechnology. The system uses NLP techniques in order to extract the necessary knowledge, such as the actors of innovations and relationships among innovations ([20, 22]).

While both of these projects focus more on analysis and related research problems, there are two projects that have created commercial tools for this purpose.

Boyack and Börner propose a way to measure certain aspects of research, such as productivity of certain papers, amount of collaboration with other fields, trends and interconnections in the field, and determination of dynamic/static sub-fields. They use data mining to cluster articles, and citations, descriptive terms, or textual similarities to discover relationships ([9, 11]). They provide a visualization tool (VxInsight) to show interesting findings; one example is the relationship between amount of funding a certain paper or sub-field receives and its productivity. Although their website [8] seems to be under construction, we have found some screenshots of their tool from another website ([10]). VxInsight has been used and proven to be helpful for gene analysis ([16]), but screenshots and explanations suggest that it is tuned more towards people that are capable of using the complex controls to gather detailed information. It provides little help to the regular researchers who would like to simply browse through trends and other researchers in their field.

SCOPUS is a tool that allows users to navigate through millions of research papers in its database. Some analysis functionalities are also available, but the main focus is to let the user search easily through papers in a database, rather than the tool inferring useful information from the data automatically ([3]). Since this tool is commercialized, using it requires a yearly paid subscription.

To sum up the related work and its relevance to our goals, we have not encountered a tool/website that provides (1) detailed visual analysis of research being done in the field, accessible to all researchers, and (2) a platform for researchers to collaborate, share ideas, and publish thoughts to the rest of the field. TRANS is specifically designed to meet these two requirements.

3 Data

Data is extracted from Transportation Research Board (TRB) web site using two different web crawlers, one to collect research need statements and the other to collect paper abstracts.

3.1 Research Need Statements (RNSs)

A research needs statement (RNS) is similar to a project proposal such that the statement briefly describes the problem to be solved, and then states the specific goals of the project. TRB is responsible for deciding whether a project will be funded, based on these statements. The webpage (http://rns.trb.org) for RNS allows users to search a previous research need within 37 predefined categories and 3462 statements, currently available for the last 4 years. After cleaning out the RNS dataset from duplicates, we generated 809 data instances, each consisting of title, description, date, and sponsor committee.

3.2 Paper Abstracts

In order to compare RNSs with published papers, we also crawled the web pages (http://trb.metapress.com/home/main.mpx) of TRB technical journal papers since 2006 and retrieved the abstracts instead of the full text. For these 9552 papers, the information of title, authors, and their addresses besides the content of abstract is also added, so that we can use this information for our visualizations.

4 Methods

In this section, we describe the methods we used to infer useful information from the raw data. Our methodology can be explained in two parts: Clustering and pairwise similarity. For both parts, first we state the motivation for using the method, then briefly describe the theoretical background, and finally go into details on how we use them to solve our problem.

4.1 Clustering

Machine learning is the theoretical basis for mining useful information from raw data. There are two main categories of machine learning techniques: unsupervised learning and supervised learning. In supervised learning, the task is to learn a concept where both data points and labels are known. In unsupervised learning, the labels are not available, therefore
the task is to find a representation of the underlying distribution ([13]). Recently, semi-supervised approaches have also become popular, where unsupervised learning is applied first (usually to a subset of the data) to assign initial labels, and then supervised learning is applied using the labels guessed in the first step ([5, 6]).

Clustering is the main method for unsupervised learning, based on the simple idea of grouping similar data points together. In our work, the goal of using clustering is to assign categories (labels) to RNSs and paper abstracts (data points). Category information in our data is not reliable, therefore we focus on unsupervised learning and clustering to assign categories. Due to space limits, our discussion of clustering is limited to its use in our methodology. For a more comprehensive overview of unsupervised learning and clustering, refer to papers [15], [17] and [7].

Two aspects of clustering differentiate between various clustering algorithms: the definition of distance between data points and the procedure of grouping the similar points together.

4.1.1 Distance measures

We can think of each data point as an \( m \)-dimensional vector, which we will refer to as feature vector for the rest of this paper. The whole data set is then a sequence of feature vectors, \( f^{(1)}, f^{(2)}, \ldots, f^{(n)} \). The similarity between two feature vectors can be defined using various distance measures. One of the most popular measures is Euclidean distance, where the distance between two feature vectors \( f^{(i)} \) and \( f^{(j)} \) is calculated as follows:

\[
d_{\text{Eucl}}(f^{(i)}, f^{(j)}) = \sqrt{\sum_{l=1}^{m} (f^{(i)}_l - f^{(j)}_l)^2}
\]

Measures that are widely used for clustering include Euclidean distance, Manhattan distance, maximum norm, Mahalanobis distance, mutual information, inner product and many others ([15, 17, 7]). Once the distance between two feature vectors can be computed, it is possible to group vectors that are similar to each other (high inter-cluster similarity) and not similar to other groups (low intra-cluster similarity). Every clustering algorithm attempts to find the ideal clustering that achieves high inter-cluster similarity and low intra-cluster similarity.

4.1.2 k-Means and Hierarchical clustering

Book chapters have been written about clustering methods and its applications, but we limit our discussion to two techniques and their application to the specific problem we want to solve.

The first technique is \( k \)-means clustering, a simple yet effective method that has been widely used for clustering. The algorithm starts with \( k \) randomly chosen ‘pivot’ data points, serving as cluster centers. Each data point \( x \) is added to the cluster of which the pivot is closest to \( x \). The algorithm is done when each data point has been included in a cluster. The main advantage of this method is its low complexity (\( O(nn) \)), however the quality of clustering depends on the initial \( k \) pivots and the number of clusters should be given by the user. There are variations where the number of clusters is discovered by the algorithm, but the best way is to test with several values and decide manually.

The second technique, or set of methods, is hierarchical clustering. In this case, the idea is to compute distance between all pairs of points, and merge the two that are closest. The same step is repeated with clusters and points, until there is one cluster that includes all points. The advantage of this method is the flexibility it gives to the user for choosing clusters, since the algorithm gives a tree (dendogram) as output. However, the time complexity is \( O(n^2) \), the algorithm is sensitive to outliers, and interpreting the tree is not always straightforward.

We use Weka machine learning toolkit ([14]) to perform \( k \)-means and hierarchical clustering on our data set. Weka is a simple Java-based toolkit that contains many machine learning tools and methods, including the implementation of several clustering algorithms. We experimented with the “Simple k-Means” implementation of \( k \)-means clustering, and the “Cobweb” implementation of hierarchical clustering.

4.1.3 Features

Due to time limitations, we only used unigrams as features, and we left our other ideas as future work in Section 6. An \( n \)-gram is a consecutive sequence of \( n \) words, therefore unigrams correspond to single words. For tokenization, the Lucene tokenizer ([11]) was applied to clean text, remove stop words, and perform truncation. For instance, transport, transportation and transporting are all truncated to a prefix that is common to all words (e.g. transport). By doing this, words of same origin are grouped together, which makes the methodology more robust.

After preprocessing text with tokenization, a weight is calculated for each term in each document (RNS or paper abstract) by applying the BM25 scoring function. The BM25 score of term \( t \) in document \( D \) is calculated as follows:
BM25 \( (t, D) \) = \frac{IDF(t) \cdot TF(t, D) \cdot (k_1 + 1)}{TF(t, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{doclen_{avg}})}

where \( TF(t, D) \) is the term frequency of \( t \) in the document \( D \), and \( |D| \) is the length of the document in words, and \( doclen_{avg} \) is the average document length in the entire collection. \( k_1 \) and \( b \) are free parameters, usually chosen as \( k_1 = 2.0 \) and \( b = 0.75 \). \( IDF(t) \) is the inverse document frequency of the term \( t \), computed using the following formula:

\[
IDF(t) = \log \frac{N - DF(t) + 0.5}{DF(t) + 0.5}
\]

where \( N \) is the total number of documents in the collection, and \( DF(t) \) is the number of documents containing \( t \) (document frequency). The intuition behind this scoring function is to assign higher scores to terms that occur frequently in a document (high TF), and rarely in the collection (high IDF). This diminishes the effect of assigning high scores to terms that are most frequent in the language, such as ‘the’, which generally do not contain any information about the document.

4.1.4 Reducing dimensionality

This approach yields 13514 unigrams, therefore the dimension of each feature vector is 13514. This creates problems for computationally complex algorithms like Cobweb hierarchical clustering. A typical approach to reduce the dimension is to remove very common and very rare words, because they contain less information than other words. To do this, we impose a lower and upper limit on the document frequency of words to be included as features. After experimenting with different values, we used a lower limit of 5 and upper limit of 400 on the RNS collection of 809 documents. This reduced the number of features to 2576.

4.1.5 Category assignment

When we apply a clustering algorithm to our data, it assigns each data point to a cluster, but the clusters do not have labels or meanings. For instance, the first instance is assigned to \( cluster_3 \), the second is assigned to \( cluster_0 \), and so on. Since we are interested in labeling each data point with a category, we need a way to map clusters to categories.

Our solution is to get an expert of the domain to do this assignment manually, for the first time, and then use those categories to perform future assignment automatically. We assume that an expert can easily come up with category names after looking at sample text and most frequent words of a cluster. We must emphasize that this manual assignment must be done only once to the initial data. Later, any document \( x \) can be automatically labeled by finding the nearest cluster center, and choosing the category associated with that cluster. If the number of clusters is \( k \), and the distance measure is \( d \), below is the formula to assign category to document \( x \):

\[
\text{cluster}(x) = \arg \min_{i=1}^{k} d(\text{center}(cluster_i), x)
\]

\[
\text{category}(x) = \text{category assoc. with cluster}(x)
\]

For hierarchical clustering, the same one-time manual assessment needs to be done for deciding on the cluster cutoff points, looking at the dendogram. We understand that this may be problematic in some cases and may not be feasible for replication, but this is the best way we could think of in the time scope of this project. Other approaches are discussed as future work in Section 6.

4.2 Pairwise similarity

We use pairwise similarity to discover repetitive research in the field, and create a “more like this” list in our website. Pairwise similarity computation is an important problem in information retrieval that deals with finding the similar document pairs in a collection of documents. It has been used in many applications such as similarity list generation, clustering objects, unsupervised learning, and retrieval of various information from a collection.

As explained in the previous section, let’s assume that each document is represented by a feature vector containing term weights. The pairwise similarity score based on this model is the cosine of the angle between the two vectors in a collection:

\[
PWSIM(u, v) = \cos(\theta(u, v)) = \frac{|u \cdot v|}{|u||v|}
\]

where \( u \) and \( v \) are feature vectors representing the documents, and \( \theta \) is the angle between these vectors. The goal is to find the top \( N \) documents that have the highest similarity score with a given document.

As the number of terms increase, the dimension of feature vectors increases, and the time complexity of tasks like clustering or pairwise similarity increases quadratically. High dimensionality is an important problem in many tasks involving operations on feature vectors. Converting feature vectors into bit sequences called “signatures” is one way to deal with the high dimensionality.

4.2.1 Generating signatures

Ravindran et al describe how to use locality sensitive hash functions to reduce the dimension dramatically, while keep-
ing accuracy high ([18]). In this approach, $D$ random vectors are used to map each feature vector $u$ from the domain $\mathbb{R}^k$ to a “signature” $s$ from a much smaller domain $[0, 1]^D$. Each bit of $s$ is determined by a dot product of $u$ and the corresponding random vector. Given $D$ random vectors $r_1, ..., r_D$,

$$s_i = 1 \quad \text{if } u \cdot r_i \geq 0$$
$$0 \quad \text{otherwise}$$

Using these signatures, the cosine similarity score between two documents can be calculated by simply counting the number of bits that differ. This count is called the hamming distance, and the cosine similarity score is calculated by the following formula:

$$\text{PWSIM}(u, v) = \cos\left(\frac{\pi \cdot \text{hamming distance}}{D}\right)$$

### 4.2.2 Computing similarity scores

Because each document is represented by a fixed-length bit sequence, it is possible to compare documents from different collections. We execute the following procedure to compute signatures:

1. For document $d$, we compute the feature vector $f$ as explained in Section 4.1.3.

2. An additional step is to normalize the vectors so that the norm of each is 1.

3. Given the number of bits $D$, we randomly generate $D$ unit vectors.

4. Each bit of the signature is computed by the dot product of a random unit vector and $f$.

5. Steps 1–4 are repeated for each document.

After generating signatures for all documents in the collection, the pairwise similarity score between any two documents is easy to calculate.

We can use this method for two purposes: First, applying this to all pairs of documents and sorting the results gives us the most similar pairs of documents. This can reveal interesting information such as repeated research, by finding very similar research statements. Cross-similarity between research need statements and paper abstracts may also reveal interesting correlations. The second purpose is to have a website feature where a “more like this” list is generated when the user submits a research need statement.

## 5 Visualization and Website

### 5.1 Visualization

In order to create a dynamic and interactive visualization, we use the Prefuse toolkit ([2]) that provides an information visualization framework implemented in Java. We have created a tool, called TRANS, that is capable of showing three different types of visualizations, using either RNSs or paper abstracts, and optionally filtering for year range and category.

TRANS contains three parts: Visualization window is located in the center, the control panel is on the upper right side, and a legend is conveniently located on the lower right side.

#### 5.1.1 Control panel

The control panel contains all of the control options of TRANS, which can be modified by the user in order to create the visualizations he/she prefers.

The first option is the visualization type, currently selected among Trends graph, bar chart, and map view. Based on this selection, the remaining options may be modified. The second option is the data type, which can be either RNSs or paper abstracts. As the third option in the list, the user can then select one of the current categories or all of them at once. The last option in the control panel is the year range slider, which can be changed to only show data published between the selected years (inclusively).

Finally, an “Update” button is included to submit the request and update the visualization based on selected options.

#### 5.1.2 Trends graph

The Trends graph focuses on showing trends of transportation research over years, in which the user can interactively find insights, such as (1) most and least popular categories of each year, and (2) categories that have become less or more interesting in time.

For this visualization, the user can specify the range of years to be displayed and the data type. Category selection is restricted to “All” for this visualization.

When the graph is created from the control panel, the upper left corner shows the number of papers/RNSs that have been loaded. The data structure we used for our visualization is a graph consisting of nodes and edges in the GraphML format ([12]). Labels on the bottom display years, labels on the left part of the graph show all categories ordered by popularity in the first year, decreasing top to bottom. Labels on the right show all categories ordered by popularity in the last selected year, decreasing top to bottom. Each node represents the popularity of a category in a certain year, and each edge
represents the change of popularity of a category between two consecutive years. Each line then represents the trend of the popularity of a category over the selected year range. The color of a line and thickness of a line is determined by the amount of oscillation using the following simple log-ratio formula:

\[
value(c) = \sum_{i=1}^{n-1} \log \left| \frac{\text{rank}_{i+1}(c)}{\text{rank}_i(c)} \right|
\]

where \(\text{rank}_i(c)\) is the rank of category \(c\) in \(i^{th}\) year, assuming there are \(n\) years. Thick and green lines show high oscillation, whereas thin and red lines show low oscillation.

Once the mouse is over an edge or node, the line is highlighted in blue and the name of the category shows on the lower left corner. When the mouse is over a node, a tooltip will pop up to show the percentage of RNSs/papers in that category, among all of the ones in that year. This allows the user to see the difference between order (position of node) and percentage (tooltip information). There is also a search box on the upper right corner which allows the user to search for keywords, so that the categories matching the keyword(s) turn blue.

Fig 1 shows the main view of \textit{Trends graph}.

5.1.3 Bar chart

The second type of visualization is a \textit{Bar chart}, in which the x axis displays a time line, and each bar shows the percentage of papers that fall into the selected category among all papers in that month. This visualization is useful to see details about trends of a specific category. When the chart is created from the control panel, the upper left corner shows the number of RNSs/papers that have been loaded.

The color of a bar is determined by the change of percentage with respect to the previous month. The color is interpolated between red and green, where green means that the percentage value has increased with respect to the previous month, and red means it has decreased.

Once the mouse is over a bar, the full date shows on the upper right corner, and a tooltip pops up to show the percentage value. The slider on the left allows the user to adjust the range of percentage values to be shown in the chart. Sliding it down will let the user focus on bars with lesser values, which may not be easily visible without doing so.

Fig 2 shows the main view of \textit{Bar chart}.

5.1.4 Map view

The third type of visualization in TRANS is the \textit{Map view} visualization. \textit{Map view} can be only used with paper abstracts, since the RNSs do not contain address information. To preprocess data, we extract zip code information from addresses using regular expressions. For addresses that do not contain a U.S. zip code, we extract the country name. We do further processing to retrieve statistics about papers in each zip code, given the year and/or category.

Once the visualization is loaded, the upper left corner shows statistics about the number of papers from U.S.A., other countries and unspecified addresses that match the selected options.

In order to make the layout look like a map, we use the latitude and longitude information of each U.S. zip code, and locate zip codes to relevant positions on the map. Each U.S. zip code is shown with a tiny gray dot. Zip codes that contain papers from the selected category and year range are shown as a green diamond, and the size of the diamond encodes the number of papers from that zip code.

There is a search panel on the lower left that allows the user to search for category (if category “All” is selected), or year (if a specific category is selected). In both cases, the zip codes that include any paper matching the keyword will be painted red.

Once the mouse is over a green diamond, the addresses affiliated with that zip code shows on the upper right text pane. The text pane can be scrolled and text can be copied and pasted for future reference. Mousing over a zip code also triggers a tooltip that contains a table of information. If all categories are selected from the control panel, then the first column of the table shows the percentage of papers from that zip code that fall into each category. The second column contains the same information for papers worldwide, and this lets the user compare local trends with global trends. The last row in the table shows total values.

Fig 3 shows how details are presented on demand in \textit{Map view}.

Zooming in may be helpful for users interested in research activity in a specific region. Scrolling up and down while holding the right click button lets the user zoom into any region of the map. In order to reduce clutter, the user can click on the “Hide zip codes” button on the lower right corner and this will hide all zip codes with no papers. The same button can be re-clicked to undo this action.

Fig 4 shows the zoom and search functions of \textit{Map view}.

5.2 Website

Our website was implemented using a three-tier architecture. At the presentation level we have hypertext markup (HTML) pages that may be viewed from the user’s web browser of choice. Our website is platform-independent, although its
Figure 1: Trends graph
Figure 2: Bar chart
Figure 3: Map view: Details on demand
Figure 4: Map view: Zoom and search
Figure 5: Web site presentation of research needs statements

presentation has been optimized for Firefox. At the application logic level we make use of an Apache web server and a Perl CGI (Common Gateway Interface) script. These components work together to generate HTML documents and serve those HTML documents to the user. They also process user arguments submitted via HTTP GET and HTTP POST commands and generate customized HTML pages in response to these user commands. At the back end, the CGI script loads and stores data from flat text files. The current web site implementation is a prototype only; were it to be adopted for professional use, data management would ideally be accomplished using a full-scale database management system such as MySQL or Oracle. With this in mind, the portions of the CGI script that involve loading or storing data have been modularized so that they may be easily replaced.

The website allows access to our Prefuse graph visualization, but also provides several additional functions:

- the listing, sorting, and searching of research needs statements (see Fig. 5)
- the ability to identify research needs statements most similar to a statement of interest
- the ability to submit new research statements
- the ability to vote on existing research needs statements
- the ability to discover users with similar interests based on their voting records and profiles.

The website also provides an alternative JavaScript-based visualization that depicts rising and falling topics among research needs statements over the years (Fig. 6). These topics were determined using latent Dirichlet allocation (LDA), an unsupervised topic modeling algorithm that characterizes topics as distributions over words and documents as distributions over topics. Because the algorithm is unsupervised, topics are not automatically assigned labels; in the visualization, each topic is represented by a list of its most probable constituent words. When a user mouses over a given topic, that topic is highlighted within the ranked list of topics corresponding to each year. This allows the user to easily identify trends in the popularity of that topic over time. Additionally, topics are color-coded using a gradated color scheme corresponding to their rank within a particular year of interest, so topics whose ranks have risen or fallen drastically relative to that year are easy to identify at a glance. Further potential refinements to the visualization are described in the section on future work.

Beyond the standard searching and sorting functions common to web-enabled databases, our website applies the NLP techniques described earlier to provide additional useful functions to users. One such function is the automatic recommendation of research statements similar to the one the user has selected, based on pairwise similarity scores. Not only will this alert the user to statements that may be interesting and relevant to his or her subfield, but it will allow for the automatic detection of duplicate or near-duplicate research statements.

When a user submits a new statement, he or she is likewise presented with a list of similar pre-existing statements in the course of the submission process. This allows for corrections and refinements to be made based on similar research, and helps to pre-emptively eliminate the problem of duplicate or near-duplicate submissions. To promote consistent categorization of research statements, the system automatically proposes categories for newly submitted statements by making use of the clustering results described earlier. Users
The web site provides a mechanism to vote for or against specific research statements, and users may view and sort research statements by their approval rate among web site users in general. To encourage active participation in the voting process, approval rates are only visible after a user has voted or specifically declined to vote on a particular statement. This restriction is also intended to prevent prior approval rates from biasing a user’s own vote. A voter may at any time access his or her own voting record, which includes statistics about his or her voting patterns over various periods of time. As one goal of the project was to allow users to discover others with similar interests, the voting record page also includes a list of other users with the most similar voting records (Fig. 7). These records may be examined side-by-side with differences highlighted, and user profiles may be consulted to determine the fields of interest and contact information of similar users.

6 Future work

6.1 Better Features

6.1.1 Using higher order n-gram features

In this project, unigram features are used to measure the similarity between documents and to find clusterings. Although using only unigrams seems to be sufficient for the current purposes, higher order n-gram models are better at capturing the use of language in text. Also, frequent word phrases such as public transportation can be used as features when a higher order n-grams model is considered.

6.1.2 Using transportation ontology

We use heuristic document frequency limits to reduce dimension, however there may be better alternatives. Using a transportation ontology to select the features will reduce the dimension by tuning towards the domain. This will preserve the information that is lost from removing very rare and very frequent words.

6.2 LDA topic presentation

The collections of words associated with unsupervised LDA topics may not necessarily be intuitive or meaningful to users. Work could be done to better communicate the meaning of the topics or how they were derived, and to assign meaningful and concise labels to these topics. It could also be informative to make use of multi-word phrases, and not just individual words, in the topic-modeling process. A transportation ontology could be consulted to determine meaningful phrases within the field.

6.3 Visualizing sub-categories

A treemap can be used to visualize the output of the hierarchical clustering algorithm. This would allow users to see levels of categorization, and view groups of papers at different levels of granularity. Another way to present sub-categories would be letting the user click through the levels of the hierarchy through bar charts. Although we have the hierarchical clustering results, we have not been able to implement these ideas and therefore leave it as future work.

6.4 Citation network analysis

A major focus of this project was the consistent categorization of documents and the automated discovery of similar documents, and citation network analysis could have provided us with an additional angle from which to attack this problem. Citation network exploration techniques such as those implemented within the iOpener workbench ([4]) could have revealed communities of similar documents, as determined by the papers that they cite and the papers that cite them. This avenue of exploration is left for future work.

7 Conclusions

We have implemented an interactive visualization tool called TRANS, that is designed to illuminate the trends within the transportation research community from different perspectives. We have also implemented a website to assist users in the discovery, submission, and evaluation of transportation-related statements of research. We believe that our work on this project can be used to make trends within transportation research more transparent to those within the research community, and to promote collaboration and informed funding.
decisions within the community. Finally, this work is intended to be used by the transportation research community, however it can be used in any research community given that the necessary data is available.

Acknowledgements

We would like to thank Michael Pack and Prof. Ben Shneiderman for their continued support and guidance throughout this project.

References


