DocTable: Table-Oriented Interactive Machine Learning for Text Corpora

Sriram Yarlagadda, David J. Scroggins, Fang Cao, Yeshwanth Devabhaktuni, Franklin Buitron, and Eli T. Brown

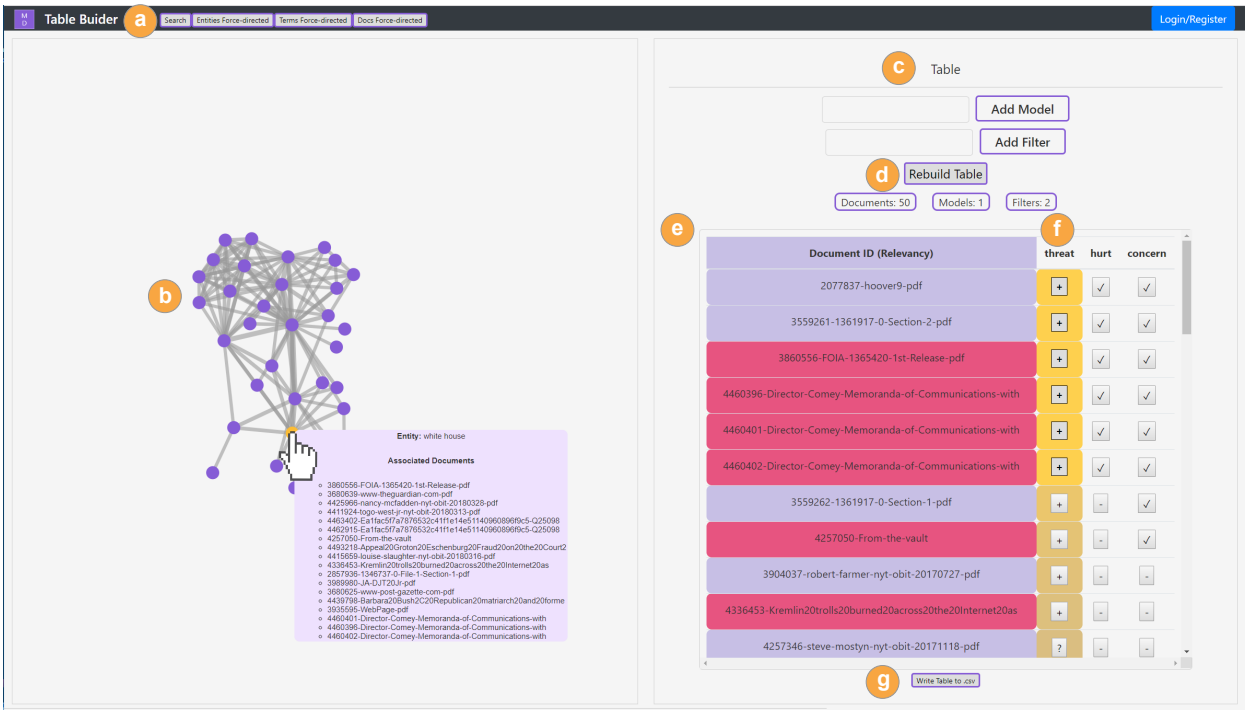


Fig. 1: The *DocTable* prototype. (a) A navigation bar to select between different visualizations. (b) A visualization pane that displays the selected force-directed visualization. (c) The table panel where users interact with model-building, including: (d) a button to trigger a model update, (e) a table view with a row per document, ordered based on relevancy to the model(s), (f) a column for each user model showing user markings or machine estimation of relevance per document, and (g) an export to *.csv* button.

Abstract— People working today with text data in any domain must develop understanding based on more documents than they can directly read. Tools exist that take advantage of visual analytics for a variety of text data tasks, including for technical experts, particularly in certain domains like intelligence and law. There is a missed opportunity to apply human-in-the-loop (HIL) machine learning to assist a general audience with text analysis tasks over large corpora that are difficult with visualization alone. In this paper, we propose an alternate approach to efficient sensemaking over document corpora, designed for users without technical expertise or training. We use a table-based interface, where the primary means of providing feedback to the machine is interactions with a data table view. This format is familiar to many, and augments the ability to tag documents into user-defined categories with machine learning that automatically predicts categories for additional documents. Marking only a few documents enables the machine learner to suggest automatic labels for the rest (with uncertainty scores for the predictions), and to reorder the table to reflect an integrated mix of the category models. Finally, interactive, force-directed layouts of topics and documents based on the models assist in sensemaking for workflow that scales beyond the practical limits of a table. To validate the technique, we present a prototype human-in-the-loop machine learning system, *DocTable*. We evaluate this technique with machine learning experiments that demonstrate the quality of the backend's response to expected user inputs, a usage scenario to demonstrate its use on real world data, and a case study of expert feedback from our journalist collaborators.

1 Introduction

Humans have always encountered the problem of analyzing information in order to make informed decisions, but in the modern world, the scale of information has increasingly outstripped human capacity to accurately and quickly discern patterns. Within this context, machine learning has received extensive attention for its ability to quickly and

- *Sriram Yarlagadda is with DePaul University. E-mail: sriram.yarla1@gmail.com.*
- *David J. Scroggins is with DePaul University. E-mail: david.j.scroggins@gmail.com.*
- *Fang Cao is with DePaul University. E-mail: fcao1@depaul.edu.*
- *Yeshwanth Devabhaktuni is with DePaul University. E-mail: yeshdevv@gmail.com*
- *Franklin Buitron is with DePaul University. E-mail: buitronfranklin@gmail.com.*
- *Eli T. Brown is with DePaul University. E-mail: eli.t.brown@depaul.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

efficiently model large datasets. Data visualization has also emerged as a powerful tool for gaining understanding. Humans remain the domain experts and the ultimate arbiters of what constitutes real insight. Thanks to the field of *human-in-the-loop* (HIL) analytics, increasing attention has been paid to integrating the human capacity for handling shifting problems and insights with the computing power of machine learners.

Inspired by an ongoing collaboration with an investigative journalism technology company, *MuckRock* (see MuckRock.com), we have developed a technique to address a need for tools for non-technical researchers to work with large sets of documents to efficiently develop insights and stories. We learned from our collaboration that journalists will often track *narrative threads*, i.e., elements of a story, through sets of documents using spreadsheets, and record information about each document as they read it. This method cannot scale to modern text corpora, so we introduce an HIL technique that aligns with spreadsheet-based workflows by anchoring interaction around a table view. Users create their own columns to represent concepts or narrative threads. As they mark relevant documents in these columns, a machine learning backend automatically models the custom topic and fills in labels for the user when possible (showing uncertainty). This draws attention to the documents most relevant to the developing narrative so users can focus on a limited range of "uncertain" documents, retrain, and iterate. To help the sensemaking process, visualizations show relationships between documents and between terms. We use the trained models in producing the visualizations so the latest user feedback is incorporated.

In this paper we (1) explain our motivation and detail the approach (Sect. 3), (2) explain the interactive machine learning techniques that enable the interactive table (Sect. 4), as well as (3) how those learned models are adapted to serve the user interface (Sect. 4). We present (4) a prototype HIL system, *DocTable*, incorporating that technology to streamline the process of developing narratives from large document corpora (Sect. 5), and validation via (4) a usage scenario (Sect. 6), a case study of expert feedback (Sect. 7), and machine learning simulation experiments (Sect. 8).

2 Related Work

This work is an application of human-in-the-loop (HIL) analytics to the longstanding problem of text analytics. In this section we discuss related efforts starting with other visual text analytics approaches and applications in industry, then provide an overview of HIL systems and some existing HIL text work, differentiating our own. Finally, we explain the relevant machine learning references.

2.1 Visual Document Analytics

There is a long history of systems for visually analyzing document collections. *Jigsaw* is a visual analytics system that uses coordinated views to enable user-driven narrative building [56]. *In-Spire* utilizes a combination of search, data tools and coordinated views to allow users to extract narratives from document collections [63]. *Analyst's Workspace* is a system optimized for large, multi-display visual analysis of document collections [5], emphasizing document and entity networks to tell stories through coordinated multiple views. *Text Insight via Automated, Responsive Analysis (TIARA)* [61] utilizes Latent Dirichlet Allocation (LDA) [9] to map the importance of document themes over time in a ThemeRiver [35] style visualization. *TIARA* also allows users to inspect sender/receiver connections via a network diagram. These systems are driven by interaction with visualizations, and though machine learning or natural language processing (NLP) algorithms may be used in creating those views, there is not an interactive machine learning component to capture interaction data to automatically improve models. Heimerl, et al., evaluate methods using a classifier and clustering with layouts of documents [36]. Hagerman, et al., use an adaptation of SVM for text with an interface for labeling documents and showing their predictions in a visualization optimized for news articles [34]. Our approach includes a table interface designed for general documents, as opposed to news articles, and organizes information in a table to manage several models at once. The labeling does not require working with a document layout, but the learned models influence the layout so it can be refined as users provide additional feedback.

2.2 Application Areas

Significant work has been done in application areas of text analytics. Within the industry of legal discovery there are an array of systems intended to streamline the process of document discovery. Most of these systems are focused on sophisticated management of large document collections rather than investigative narrative building. A typical work-flow involves: (1) data ingestion, (2) manual tagging and batching of documents, (3) usage of information retrieval techniques to locate tagged and batched documents in order to cross reference with other collections. Typical examples of these systems include *Everlaw*, *Logikcull*, and *DISCO* [18, 26, 45]. Some systems such as *RAVEL* incorporate sophisticated coordinated views oriented toward exploratory visualization [44]. Products such as *Brainspace* utilize interactive visualizations as data ingestion techniques to narrow down document collections, then allow users to apply machine learning techniques to discover additional documents [12]. Their interaction paradigm asks users to tag documents as responsive or non-responsive, and uses a proprietary algorithm called *Continuous Multi-Modal Learning* to help improve sorting. Instead we use metric learning, allowing us to leverage labels efficiently and provide a flexible interaction and re-ranking mechanism based on multiple user-created models simultaneously.

In journalism, there are more tools for analytic tasks like managing documents. *DocumentCloud* [21] is primarily an organizational system for documents that have been processed using Thomson Reuters' *Open Calais*, a service for extracting semantic metadata. It offers journalists search functionality and entity visualization, as well as tagging and annotation capacity. *Overview* [50] is designed as a complement to *DocumentCloud*, offering users advanced interactive visualizations generated via clustering algorithms as well as an API for writing one's own visualizations. These approaches stop short of leveraging machine learning to facilitate the rapid comprehension of the documents they help to manage.

2.3 Human-in-the-loop Analytics

Human-in-the-loop (HIL) analytics is a field devoted to getting the best out of both humans and computers by facilitating their cooperation[1]. For a deep discussion of work in this area and the ways human and computer work together, we defer to frameworks by Keim et al. [42], Sacha et al. [52], Endert et al. [25], and Brown et al. [14]. Essentially the point is to use computers for their best attributes and humans for theirs, combining raw computational power with the ability to develop novel insight and adapt to shifting problems and goals.

Many HIL systems have been built, and they tackle a variety of problems. There are different approaches, including explicitly allowing interaction with machine learning algorithms, e.g. PCA [39], regression [47], and decision trees [59]. Another approach is to have the user interact directly with a visualization of the data, in a way that matches the semantics of their understanding as a domain expert, i.e., via *semantic interaction* [23]. This approach has been used for a variety of domains including some outside analytics like image search [29] and social group discovery [3]. *Dis-Function* is an example system for generic numerical data that gives a projection of the data and allows a user, by manipulating data points directly (observation-level interaction [24]), to provide feedback on their relationships that is used behind the scenes to update a model. This mechanism is used iteratively to improve the model, leading to a better projection and an understanding of what data features are important [15] (additional work on this general problem includes multiple techniques [24, 43, 54]). Other example problem spaces tackled with semantic interaction include ranking [60] and network alarm triage [4].

There are tools using semantic HIL for text analytics. *NLPReviz*, is built for clinical text data, i.e., electronic medical records. A user has multiple columns of separate labels to fill out about the content of different patients' records, e.g., particular medical conditions. The system uses support vector machines to attempt to fill in gaps, and

[1] *HIL* focuses on interactive analytics but is nearly interchangeable with the related, broader fields of *interactive machine learning* and *human-centered machine learning.*

provides tools like visualizations of text content, model uncertainty and label distribution to aid a human in fixing the predictions [58]. Though this sounds similar, we are providing the user with the ability to create their own ad-hoc models of generic text data and use a learning mechanism more suited to a small number of labels. *ForceSPIRE* [23] is designed for sensemaking in textual data. It utilizes user interaction with the spatial representation of data to steer the backend which learns from these interactions and updates the spatial representation. Users can implicitly re-weight entities via four semantic actions: repositioning documents, highlighting text, search, and annotating. Bradel, et al., extend this to use multiple models directly in the spatialization of documents [11]. This concept was flipped in a visual analytics system that creates a spatialization of the terms instead of the documents [22]. Further work in the area includes a term spatialization with semantic interaction of backend model learning based on moving words [13, 16]. That HIL term spatialization is related to this work, except that our user experience and interactive learning focus on a table instead of a spatialization, and we use a different backend learner.

With *DocTable*, we are working with unstructured text data, and building a user-specified set of models with the intention of helping to develop a narrative. The user creates new models to represent concepts on the fly and defines them by giving positive and negative examples from the text. There is no ground truth label for any documents (as with, e.g., diagnosis). Instead we allow the user's guidance to steer the model-building, focusing on capturing their mental model of the problem and providing recommendations accordingly. To this end, models built by users are used to sort documents to help the user find the most relevant material to develop their narrative. In addition, using this type of model or collection of models to customize a visualization of terms requires special consideration explained in Sect. 4.3.

2.4 Machine Learning

The machine learning approach adopted for *DocTable* is metric learning. Metric learning does not fall neatly into either main camp of machine learning (supervised or unsupervised), but it is well-suited to interactive contexts. There are multiple variants of metric learning and it can be run with fully labeled data (as a supervised technique), but a key strength is its semi-supervised ability. By taking constraints rather than labels, typically of the form "x_i and x_j should be very close together (or far apart)", metric learning can be flexible. Metric learning has been used for ranking [46], retrieval [37], face recognition [17, 31], and HIL systems [6, 15] among others uses (see metric learning survey [65]).

Metric learning can get substantial benefit from a small number of labels [64], making it all the more suited to interactive contexts where maintaining user attention is difficult. A downside is that in interactive systems, any analytics has to be run quickly, and metric learning is not typically cheap because in many formulations, all constraints are used every time the model updates. Online learning algorithms work well for situations needing a quick update to a model based on new information. Fortunately, there is an online metric learning algorithm, LEGO [38]. We adopt LEGO for this work, using it to quickly turn around model changes as users provide small incremental amounts of labels.

3 Motivation, Design and Approach

In this section we explain the motivating problem and how our design goals evolved from discussions with a collaborator in journalism technology. The main contribution of this paper is an interactive machine learning technique, but our inspiration and part of our evaluation is motivated by the challenges of this domain. We finish the section explaining how our overall HIL approach addresses these challenges and design considerations.

3.1 Motivating Problem

This work is inspired by collaborators at a not-for-profit journalism technology company, *MuckRock*, that supports journalists by gathering large collections of public documents, often by assisting in filing Freedom of Information Act (FOIA) requests to government agencies. They make millions of pages of documents accessible with tools for citizen journalists, with a goal of increased accountability. In talking to our contact, Managing Director Michael Morisy, we iteratively developed the design by discussing drawings and diagrams and learning about journalist workflow. The idea for the table component that anchors this work came from the revelation that typically, journalists currently rely on Excel. This situation surprised us, since the workflow they described involved creating a spreadsheet with columns for topics of interest, and manually filling columns as they read documents. While large newsrooms may have access to fancier tools, these spreadsheets could be used by anyone, with flexibility to have columns include tags or even extracted values like the number of people involved in an incident to be used as data in later analysis.

Our collaborator also provided us with a series of user stories to summarize the challenges faced by investigative journalists. These uncovered some of the competitive pressures faced, especially by smaller news outlets or independent journalists. One unifying feature of these stories was that, regardless of technical orientation, these investigative journalists still rely on manual review of documents and generally use spreadsheet software for analysis. Another insight for us was that this particular audience is perfectly comfortable with complexity and nuance, just not well-versed in technology. Specifically, they were happy to engage with the details of abstract representations of their work, as long as the specific technical details are not essential to using the tool. We hypothesized that the ability to work through many documents without the human capital available at a large organization could be answered with interactive machine learning tools and began to iterate design ideas. From these conversations we developed a clear set of goals for an HIL approach:

- G1 Enhance scalability over the spreadsheet baseline by assisting in quickly organizing documents around user-defined themes.
- G2 Use a familiar, table-oriented interface to reduce the learning curve.
- G3 Maintain access to document materials for specific review.

The HIL approach explained in the next subsection, and implemented in the *DocTable* prototype, addresses these goals well enough that the collaborator has begun integrating the technology into their own user toolkits on their website.

3.2 Human-in-the-Loop Approach

Conceptually, we approach the challenge of developing a narrative from a corpora of documents as a process of iterative interaction assisted by model building and visualization. In this approach, we start with a set of documents retrieved from the overall corpus via keyword search. The user then begins the human-in-the-loop sensemaking process and iteratively uses the visualizations and document content viewer to learn about the documents, provide feedback through the table, and update the model to get a renewed view of their data.

The loop proceeds through the following steps:

1. Documents are listed in an interactive table ordered by the backend machine learning model (at first, simply based on order in the original search).
2. Force-directed layouts provide the user with overviews of keywords and documents and their relationships based on the model (once it is available).
3. The user creates table columns to represent concepts in their sensemaking. With help from the visualizations and the table, the user marks a subset of the documents as either relevant or not relevant to their concepts.
4. This interaction generates an update of the backend metric learning model. The new model is used to label the rest of the table rows relative to the user concepts (showing uncertainty), and update the relationships between documents and between terms that are shown in the visualizations to reflect the new user feedback.

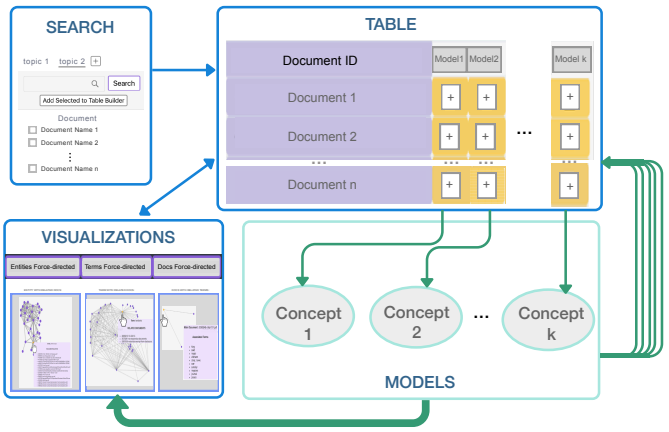


Fig. 2: This diagram shows the human-in-the-loop approach in *DocTable*. The blue-outline items are interface components and interactions between them are indicated by blue arrows. The green box represents the models learned on the backend. Green arrows show the flow between the models and the visual components, with each user-defined concept (table column) controlling its own model. The corresponding workflow is explained in Sect. 3.2.

5. The process repeats iteratively.

The feedback loop can be seen in Fig. 2, which shows the flow of information from user to models and back again. The user continues this process until they have extracted the desired information about their documents.

4 Using Metric Models in Tables and Spatializations for Visual Text Analytics

To fulfill the HIL approach of Sect. 3.2, we must address how to learn from user feedback with the table view, and then how the new models can be used to update the table and visualizations.

4.1 Table Interactions into Metric Learning Constraints

The table interface has one row per document and only the columns that the user specifies. The user creates as many columns as they like, each one representing a concept in their sensemaking. (We ignore the other type of columns in the prototype (see Sect. 5) in this section as they do not contribute to the model learning.) For example, for a journalist looking at public spending on events, their columns could represent concepts such as if a document is about a local event, or discusses the budget. Using the visualizations and reading documents, the user is able to judge the relevancy of each document to each concept, and check a box in the table indicating a *positive* or *negative* relationship. This relevance feedback is then used to learn an updated model using a metric learner.

Metric learning is a good match to this application because it works well with small amounts of feedback (see Sect. 2.4). Commonly, metric learners use constraints in the form of tuples, e.g. (i, j, `together`), meaning points i and j should be the same class or close together. To convert from user labels to constraints, we assume that, within each user concept (column), documents marked as *positive* are similar to other documents marked as *positive*, but dissimilar to documents marked as *negative*. Each concept's model is trained separately with the following constraints for the metric learner: (1) a `together` constraint for all pairs of documents the user checked *positive*, (2) a `separated` constraint for all pairs where one document is marked *positive* and the other *negative*. Notably absent are constraints for pairs of documents sharing a *negative* label. This is because there are many ways that a document could be irrelevant to a concept and the user semantics of the *negative* marking should not be interpreted by the learner as feedback that the *negative* examples are similar to each other.

4.2 Updating the Table With Metric Models

The new models must now be used to improve the table, specifically through re-ordering the documents for relevance, marking them with relevance predictions, and annotating them with the model's degree of certainty in its labels. The metric model is represented in the backend by a square matrix of size equal to the number of features used to represent the documents. Since even a relatively small document corpus is likely to have a large number of terms, we do not use the terms directly as our features as that would make the table update process slow. Instead, we use feature extraction to reduce the feature space by applying Principal Component Analysis (PCA) [40] to obtain a manageable number of features p. The distance between any two documents is the Mahalanobis distance between them using the metric model's matrix A ($p \times p$) as[2]:

$$D_A(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} - \mathbf{x_j})A(\mathbf{x_i} - \mathbf{x_j})^T \tag{1}$$

Pairwise distances are useful in many contexts, but we need more flexibility to incorporate the model into our table and visualizations, so we use a factorized version to transform the data with respect to the model. Using the Cholesky factorization, a matrix A forming the basis of a metric learning model can be factored into the product $A = L^T L$ of a lower triangular matrix with itself. It is equivalent to calculate distances between points using A or first multiply data by L and then use regular Euclidean distance [62]. This equivalence can be seen thanks to the way that transpose is distributed over matrices, i.e., using the fact that $(XY)^T = Y^T X^T$ for Equation 2.

$$\begin{aligned} D_A(\mathbf{x_i}, \mathbf{x_j}) &= (\mathbf{x_i} - \mathbf{x_j})A(\mathbf{x_i} - \mathbf{x_j})^T \\ &= (\mathbf{x_i} - \mathbf{x_j})L^T L(\mathbf{x_i} - \mathbf{x_j})^T \\ &= \left((\mathbf{x_i} - \mathbf{x_j})L^T\right)\left(L(\mathbf{x_i} - \mathbf{x_j})^T\right) \\ &= \left(L(\mathbf{x_i} - \mathbf{x_j})^T\right)^T\left(L(\mathbf{x_i} - \mathbf{x_j})^T\right) \qquad (2) \\ &= D_I(L\mathbf{x_i}, L\mathbf{x_j}) \qquad (3) \end{aligned}$$

The result, in Equation 3, replaces A with the Identity, I, because the model has already been included. Using the matrix L, we transform the data such that distances between points are equivalent to the Mahalanobis distances, i.e., the data themselves now reflect the model.

To order the documents in the table, we calculate a mean document vector of only the documents marked as relevant by the user. As a surrogate for its relevance to the user's concept, we calculate each document's distance to this mean vector. We also calculate the percentiles of the documents' distance to the mean. Documents in the twentieth percentile are labeled as relevant (+), documents in the eightieth percentile as irrelevant (-), and all others as uncertain (*?*). The front end uses these percentiles to order the documents, stably sorting based on the mean percentile of each document across all model columns. Documents directly marked by the user as relevant or irrelevant are given percentiles of 0 and 100 respectively to ensure they are always pushed ahead of model-based recommendations. The consistent assumption of *DocTable* is that the user knows better than the machine learner what document are relevant, and therefore their choices are never overridden, and always given priority over model results.

4.3 Using a Metric Model to Visualize Relationships

Another contribution of this work is how we combine multiple models from user interactions to visualize data relationships in a customized way. The document-document relationships are relatively straightforward, but the term-term relationships require a novel approach.

Each document is represented as a vector in the PCA-reduced feature space, $1 \times p$, transformed by the matrix L (see Sect. 4.2). The document visualization uses pairwise distances to produce the edge weights of a graph of all documents that can be visualized with, e.g., a force-directed layout with a user-controlled threshold for edge strength.

[2]The square root is left out of the calculation for efficiency without consequence because we are using these values only for comparison.

The calculation is not as straightforward for relationships between terms, however. Documents are represented for computation as coefficients of the principal components from PCA, meaning the components are acting like a set of terms, which we refer to here as PCA-terms. Because the documents, and thus the metric model, are represented in the PCA space, their direct correspondence to the original terms is lost. Additionally, visualizing all the terms would overwhelm the visualization, so we must reduce the set in the process.

In picking an appropriate subset of terms, we want to take advantage of the learned model, so we start by subsetting the PCA-terms, which are directly referenced by the model. We select a subset, size $p' < p$, of the entire PCA-term vocabulary of the corpus by thresholding based on the weights on the diagonal of metric matrix A[3]. The ideal threshold will depend on the data and can be determined empirically, or by using a heuristic like a percentile.

Next, we consider the principal components matrix, P, which contains the relationships between the original terms and the PCA-terms. Specifically, P is the $m \times p$ matrix, where m is the number of original terms and p is the chosen number of components, used to transform the data to PCA space. For each original document vector $\mathbf{x} \in \mathbb{R}^m$, we get the elements of PCA-space vector $\mathbf{x}' \in \mathbb{R}^p$ via the dot product with each principle component, $\mathbf{x}'_i = \mathbf{x} \cdot \mathbf{p}_i,\ i = 1, \ldots, p$. This formula is commonly written in matrix form as $X' = XP$, for X the $n \times m$ matrix with all documents as its rows.

The rows of P represent the original terms, and the matrix contains their relationship to the PCA-terms. We can add the weighting from the model to this matrix to create the matrix W. Let vector $\mathbf{a}$ be the diagonal of matrix A, the weights of all PCA-terms. First, use the weights in $\mathbf{a}$ and apply a threshold to remove the elements for the least relevant PCA-terms from $\mathbf{a}$ and corresponding least relevant columns from P. The trimmed versions are $\mathbf{a}'$ and P', with p' instead of p elements and columns respectively. Apply the weights for each PCA-term from $\mathbf{a}'$ to the columns of P' to create:

$$W_{ij} = \mathbf{a}'_j P'_{ij} \tag{4}$$

Note that this cannot be done simply by matrix multiplication with A because of the dimension mismatch from trimming terms. In W, each row is the representation of an original term based on a subset of the principal components.

Finally, we must trim the set of terms, so we threshold again, now based on the row values of W. We compute a relevance score for each term:

$$\omega_i = \sum_{j=1}^{p'} W_{ij} \tag{5}$$

The terms with the top scores are selected (the ideal threshold will again depend on the data), giving us τ weighted term vectors $\mathbf{t_i} \in \mathbb{R}^{p'}$. We use Euclidean distance to calculate the pairwise distances, $dist(\mathbf{t_i}, \mathbf{t_j})$, for visualization.

4.4 Using Multiple Models

The interactive table allows the user to generate multiple models, but we do not want to complicate the interface with features to mix and match subsets of models applied to the visualizations. The models being created should all contribute to the same investigation and can work together. To account for all the models from the table, we first combine their matrices into a single metric that represents all in conjunction. Averaging the matrices is not appropriate as we instead want to combine the effects of the transformations of the data. Fortunately, as described in Sect. 4.2, the matrices are used in a factored form to process the data. We can stack these linear transformations for multiple learned models. Specifically, we use the factored versions of each of the k user models, $A_i = L_i^T L_i$, $i = 1, \ldots, k$. By multiplying the L_i together, we can use the usual version of the distance function (Equation 3), substituting $L = L^*$:

$$L^* = \Pi_{i=1}^{k} L_i$$

[3]The off-diagonal elements of the matrix may improve distance calculations, but the diagonal alone is often good enough and is easier to interpret [64].

Fig. 3: Document Viewer. When a PDF version is available it is used for readability, otherwise this modal dialog contains raw text. The top portion contains extracted entities, if available.

5 The DocTable Prototype

DocTable is a prototype implementation of the techniques discussed in Sect. 4. There are three primary components: the search pane, the table pane, and the visualization pane, plus a modal document viewer. With these controls, users can find documents, create models of topics to help build a narrative thread, and use visualizations that incorporate those models to improve their understanding. The following sections detail each of the main components, including interface features and useful information about their implementations. Sect. 6 showcases the core functionality of *DocTable* in an order that reflects the expected workflow for users.

5.1 Initial State and Search

New sessions in *DocTable* (see Fig. 1) load with an empty search pane[4] and table pane (Fig. 1 (c)). The search pane is the starting point for all sessions as it allows the user to get a relevant subset of the whole document corpus. After searching, users are given an ordered list of results of the 50 documents most relevant to their query. Once users have a list of results in view, they can click on an individual result to render the document in the document viewer (see Fig. 3). Users can either manually review documents in the document viewer and add them individually to the table or batch add all of them to the table. They may repeat this until they have found as many documents as needed.

Behind the scenes, as part of pre-processing the text data, a *Document Term Matrix* (DTM) is constructed before the user's session begins. Using normalized token-frequencies via the popular *term frequency-inverse document frequency* (TF-IDF) weighting approach [7], this matrix represents which terms from the text are in which documents. For a search, the query's *term-frequency* vector is prepared and multiplied by the DTM to get a relevance matching score for each document in the corpus. Search results are ordered by this score. In the current prototype of *DocTable*, we use a corpus of documents our collaborator retrieved from a Freedom of Information Act (FOIA) request to the FBI. The resulting DTM has 13,471 documents and 84,169 unique tokens.

5.2 Document Viewer

DocTable includes a document viewer (Fig. 3) which can be accessed from both the search and table panes. On clicking a document, the user gets a scrollable modal window. The document viewer also functions as a second method for adding documents to the table. If available with the data, the viewer's header includes a series of buttons displaying entities associated with the document. Clicking one adds relevant documents to the table.

5.3 Table Pane

The core component of *DocTable* is the table pane (Fig. 1 (c)). Once a user has added documents to the table, they are presented with two options for creating columns: *Add Model* and *Add Filter*.

[4]The search pane is not shown for space reasons, but it is straightforward. It provides tabs to keep track of multiple search results, and the ability to add results to the table.

For a filter column, the user specifies a string, and the column automatically indicates with a check mark if a document (row) contains the specific term. These columns are intended as a simple tracking device to spot documents in which specific words or phrases occur. Model columns (Fig. 1 (f)) do not match based on the specific phrasing. The provided string is an arbitrary name supplied by the user to represent the narrative thread they will associate with that column. Read cross-column, the model column names compose a story of multiple topics.

When a model column is added, the values default to the *?* symbol, indicating uncertainty regarding relevance. Using the filter columns to spot initial documents to target, users then begin to manually review documents. Users indicate document relevance to a particular column (i.e., narrative element, theme or topic) by clicking on a cell to update with the symbol - ("not relevant") and the symbol + (indicating "relevant"). Once the user has reviewed a certain number of documents (currently set at 5), *DocTable* prompts the user to retrain the table. However, users are able to retrain whenever they desire with the button at Fig. 1 (d).

The table retrains based on the user's manual classification, indicating on a per-model basis documents the model believes to be relevant, irrelevant, and uncertain. User inputs are never overridden by the model. On retraining, the table is re-ordered by the aggregate relevancy of a document to the entire set of models. User inputs are given higher priority than model updates and documents relevant to more models will sort highest.

In addition to re-ordering the table by aggregate relevancy, table updates employ a number of visual semantics to guide navigation of the table (see Fig. 1 (f)). User inputs are in bold font to distinguish manual review from recommendations made by *DocTable*. Additionally, cells are color coded to indicate the model's current certainty with darker gold indicating stronger relevance and darker blue indicating stronger irrelevance. The color space between these two endpoints is generated from a CIELAB color space interpolator [19].

After retraining, users can then focus on documents recommended by the model (either confirming or rejecting the model's recommendation) and retrain the table. Users can also export final results to *.csv* format (Fig. 1 (g)). The updated view is focused on enabling users to quickly iterate through documents resulting in a table completely sorted by relevance to their entire narrative. The table still enables a traditional work-flow of manual inspection of documents but by responding to minimal user interaction in order to update document's relevancy, *DocTable* allows users to focus on documents with uncertain relevancy, rather than manually reviewing every single one.

5.4 Visualization of Term and Document Relationships

There are three force-directed graph visualizations available, showing relationships between entities (if provided), terms, and documents (Fig. 1(b)). They all have the same essential features. Dragging and dropping the points pins them, making it possible to explore the relationships as the force-directed simulation resettles. Clicking a point marks it yellow and causes relevant documents to be highlighted in the table viewer with color. Double clicking resets the pinning and table highlighting. Mouseover text shows detail about the item. For documents, the terms are shown. For terms and entities, documents are shown. The entities view is shown in Fig. 1 and the documents view is in Fig. 4. In Sect. 4 we explain the details of calculating the distances between terms and documents. Using those distances, the visualizations are a straightforward implementation in D3.

5.5 Implementation Details

DocTable is built on top of the Flask micro web framework [28]. The interface is implemented with React.js [27] and D3.js [10]. AJAX requests are sent via jQuery [57]. The backend model is implemented in Python 3 with primary dependencies on numpy [49], scipy [41], Gensim [51], and nltk [8]. The application was deployed as an internal tool for our collaborator on an Amazon Web Services EC2 [2] instance using NGINX [48], Supervisor [1], and Gunicorn [32].

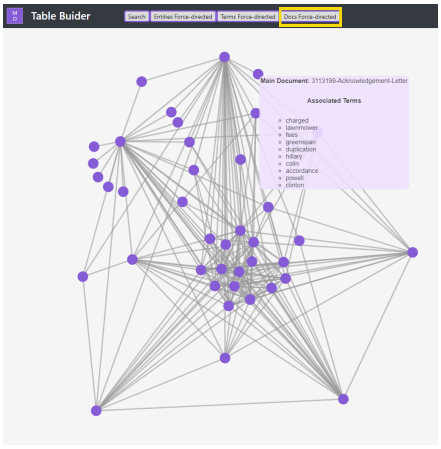


Fig. 4: Document Force Directed view, showing relationships between documents and the important terms in each document in a tooltip.

6 Usage Scenario

As described in the previous sections, one of the most important and interesting uses of this tool is in finding hidden nuggets of data about a particular theme. As we put this tool into action on a database of FBI files on individual Americans, provided by our collaborator, we started with the idea of better understanding how two important Federal agencies, the FBI and DEA, interact with each other. We uncovered some key information which otherwise would not have been easy to find.

Based on our understanding of FBI history, there were times in the past (mainly during the Clinton administration) when there were proposals to merge the DEA and FBI. We wanted to find documents in the database that relate to this theme and extract some interesting insights. After an initial search using the phrase "dea fbi merger", we had an initial set of documents to interact with. After reading through several, we marked a few as relevant and a few as non-relevant and ran the metric learner to reorganize the document set.

We then moved forward and started interacting with these documents through the force-directed graphs. In particular the term-to-term force directed graph seemed most instrumental in this particular exploration. By moving around various terms in the graph, we were able to find those nodes that were the most influential – determined as a function of how many other nodes connected to them. While most of these "influential nodes" were simply terms like "New York", "Washington", and "Clinton", which were expected given the subject matter, we noticed an interesting term – "Heriberto". As we can see in Fig. 5, "Heriberto" was a term connected to many other terms making it quite influential and central to the document set we were exploring.

Upon searching the internet for "Heriberto" we quickly learned that it is the name of a notorious Mexican drug lord who was active around the time of the proposed FBI-DEA merger. He was somehow related to the merger, which was an interesting find. Through *DocTable*, we were able to quickly uncover this name, which was otherwise buried in a massive stack of documents about the era. Reading through that pile, Heriberto may not have stuck out at all, but with a term-term force-directed layout based on the custom models about the FBI-DEA topic, he quickly surfaced. Furthermore, since the force-directed graph lists out important documents associated with this term, we know the reference material that is our starting point for understanding Heriberto's role.

7 Domain Expert Feedback

We deployed our *DocTable* prototype to get feedback from journalists. Four employees of *MuckRock* used the tool with one of their datasets, a corpus of 13,471 documents from Freedom of Information Act (FOIA) requests to the Federal Bureau of Investigations (FBI). There were two men and two women with ages ranging from mid twenties to mid

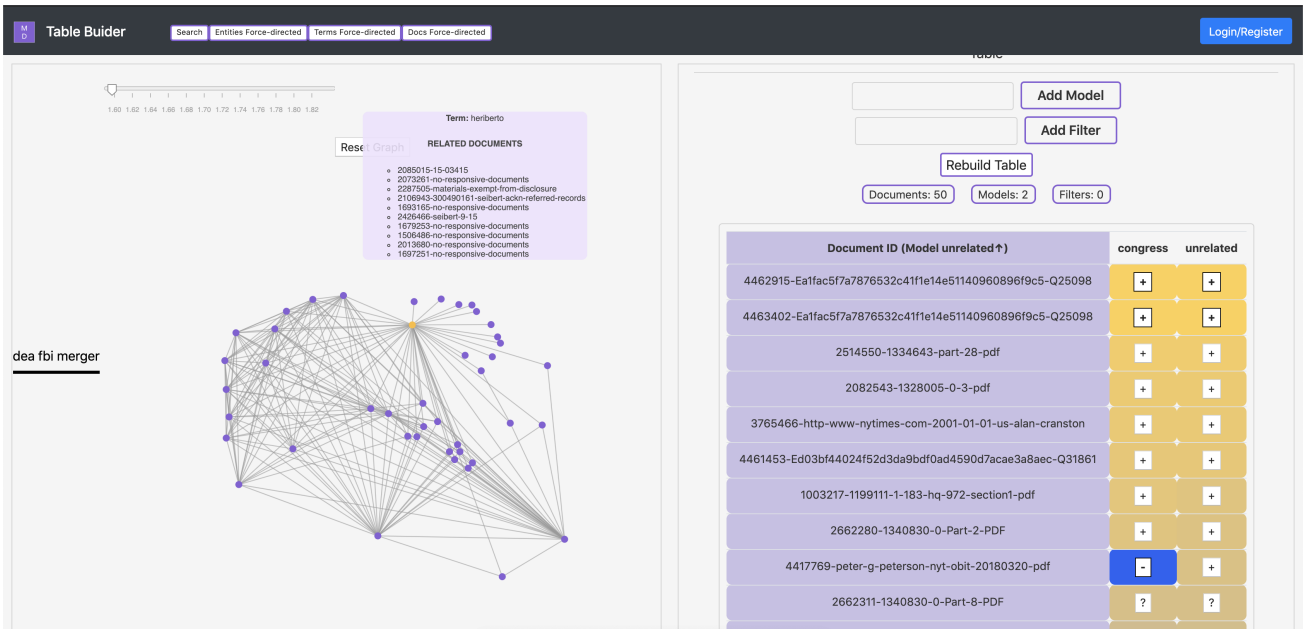


Fig. 5: This view of *DocTable* during the usage scenario of Section 6 shows the table view with two positive document labels by the user in each of the two model columns, and one negative label in *congress*. The visualization on the left is showing relevant terms, and leads us to discover Heriberto, who is the only one connected to quite a few keywords.

thirties and experience in journalism ranging from about 5 to more than 10 years[5] (median 7.7). Though some participants had done advocacy or engagement work with artificial intelligence in journalism, none have technical background in the subject. Some participants had seen a previous prototype of a component of *DocTable* about six months prior, but this evaluation is a case study, not a user study [53], so there is no task performance analysis to confound. Each expert was given a tour of the prototype and asked to work with the tool on their own open-ended investigation of the data

The visualizations in *DocTable* immediately highlighted some features of the data that the subjects were already aware of, like the fact that many documents are actually email responses to FOIA requests, as opposed to content provided by government agencies. For example, a common type of document is a *Glomar response*, in which an agency says they acknowledge the request but "can neither confirm nor deny" the existance of the requested material. Names that are involved in requests are often tagged as important terms in the visualization, especially "hardy", representing David Hardy, Section Chief of Record/Information Dissemination Section (RIDS). After discovering these issues, participants would select only documents that were not response emails, and results got more interesting. In this section, we break down the results by the main components of the tool (table builder and visualizations), and then discuss some of the overall impressions and understanding of machine learning in journalism.

The table builder was well received. One participant, asked during the survey if the predictions it made were accurate and if they improved with additional labeling said, "Yes and yes... Pretty accurate, pretty quickly. Used it a few times and it makes a ton of sense to me now." The participant noted that "how quickly it was able to relatively smartly start arraying the data based on not that many entries was really good." The fact that it showed uncertainty was also appreciated.

The reception of the visualizations was less uniform. As noted in Sect. 3, our design was an attempt to push the journalists toward a more scalable approach than tables. The participants had not worked with force directed visualizations, so they were figuring out the interaction as well as the semantics at the same time. One participants first reaction was that it was "more theoretically useful than useful." Another participant understood the point and the interaction immediately, explaining it was "my favorite feature, seeing the ties visually, ... I think that was the most useful thing, just seeing that visualization. I'm a visual learner."

Initially, the documents view was crowded and the participant did not see the benefit, but with some interaction and explanation, they looked at the terms associated with a group of documents and had a moment of insight, "Oh, it's all Comey memos. Okay, that makes sense... could be really useful." Another subject preferred the documents view noting it was "helpful to see the ties instead of going through the documents one by one." Of the term-term visualization, one participant observed the terms were "mostly people who are involved." We were happy to see that the model was automatically emphasizing the relevant people, in line with our expectations given the simulation experiments described in Sect. 8.2. In the end, the collaborator was impressed enough to devote resources to integrating the technology into their public web platform, and anticipate making the functionality live by 2022.

8 Quantitative Evaluation

The usage scenario (Sect. 6) and expert feedback (Sect. 7) demonstrate the value of our approach, but we evaluate the effectiveness of the underlying machine learning algorithms for this interactive context with machine learning experiments to get a quantitative analysis.

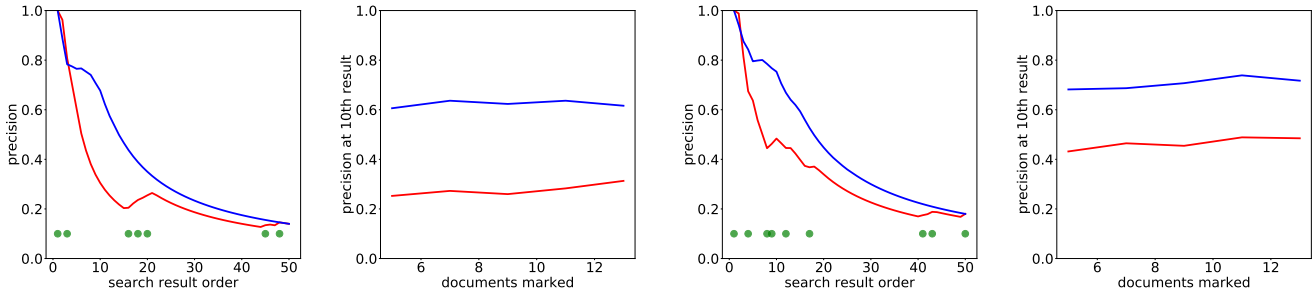
8.1 Evaluation of Interactions' Effect on Models

This evaluation quantifies the ability of *DocTable* to bring the most relevant results to the top. We used a programmatic, experimental design on a labeled document corpus, where the label (or category) of the document is used as a surrogate for the concept that the user is attempting to find. Using labels to simulate human interaction is a common evaluation tool in areas of machine learning where model building is dependent on human feedback, like active learning [55]. It provides a consistent way to estimate user behavior, so we avoid noise in that basic element of the experiment. In this approach a label is chosen to be the topic of interest, and the relevancy of a document is solely based on if its label matches.

In these experiments, we simulate iterative interactions and see how they affect the document ordering in the table. We compare the ordering produced by *DocTable* to the document ordering generated when the user simply moves relevant documents to the top. Multiple experiments were conducted using different search terms and the performance was

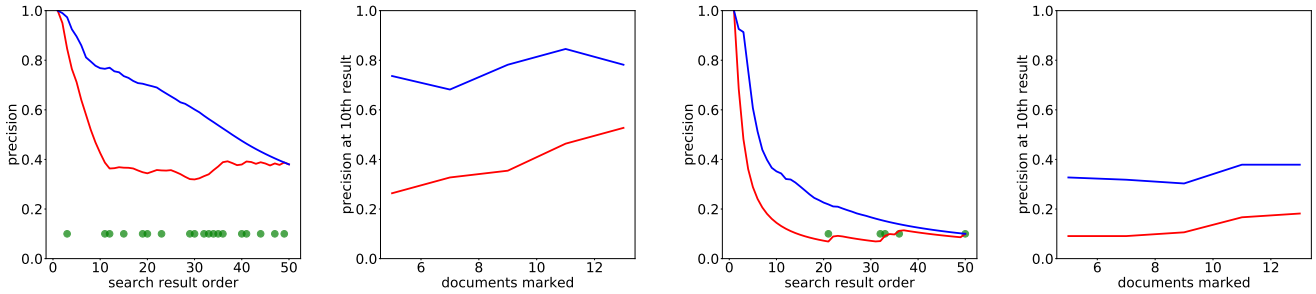
[5] Given the size of the company, giving exact numbers of years could potentially identify individuals.

Concept: vegetable oil. Search terms: “food soy fats soybeans vegetable” on the left and “food soy fats” on the right.



(a) (b) (c) (d)

Concept: Foreign currency exchange. Search terms: “global market currency” on the left and “global” on the right.



(e) (f) (g) (h)

Fig. 6: This figure shows the results of the quantitative evaluation (see Sect. 8.1) of the *DocTable* mechanism to evaluate how it improves the order in which documents are presented to the user in the table. There are two different target concepts, *vegetable oil* on the top row and *foreign currency exchange* on the bottom. For each concept we have chosen two sets of search terms. For each search term, there is a pair of graphs. The left graph of each pair (a, c, e, g) shows average *precision* across experimental runs versus position in the simulated table results (also called the search result order in reference to information retrieval). The right graphs (b, d, f, h) show how the number of labels provided by the user impacts the quality of the table results, using *precision-at-n* (for $n = 10$). **Legend:** The red line is the performance of putting positively labeled examples at the top of the list, and the blue line represents *DocTable*'s performance. The green dots on the *precision-at-n* graphs show at what index in the original search result the positive labels were located (the vertical axis value is constant).

evaluated with *precision*. Precision, which is the ratio of correctly predicted positives to the total number of possible positives, ignores the model's accuracy at predicting negatives. It is commonly used in information retrieval for search results. When used to evaluate an ordering, like the results of a search query, precision gives the proportion of the results that are relevant. Because the top results of a search are most important, *precision-at-n*, where precision is calculated based on the top n results, can be a more focused measure. Our comparisons between baseline and the *DocTable* approach over multiple datasets and random trials provides an understanding of how the combination of an online metric learner with user interaction, rather than user interaction alone, can result in a better table view.

8.1.1 Methodology

We chose the ModApte subset of the Reuters-21578 dataset [20], a collection of 10,766 news articles, each belonging to one or more of 90 categories. Each experiment broadly involves (1) identifying a small set of search terms that are deemed relevant to a document category, (2) retrieving search results for a combination of these terms as if they were used as a query in *DocTable*, (3) simulating user interaction with the resulting document set by marking a subset of these documents as relevant or non-relevant, and (4) comparing the variation in the precision of the search results before and after learning is applied.

The search terms related to a concept were selected by manually scanning the overview section in the associated Wikipedia article (chosen words are shown in Fig. 6). This mimics a user's approach to finding documents with *DocTable*, where they may not have a full picture but, instead, a general idea of the kind of documents they are trying to uncover. Next, we assume the user batch adds all documents from search and uses *DocTable*, applying correct labels to a random subset based on the ground truth from the data.

We run experiments with multiple searches and multiple sets of documents. We generate all possible combinations of the selected search terms and use each combination as a query to evaluate. For each query, we take the first 50 search results and randomly pick a subset to label. We test subsets of sizes 5 documents to 15 documents in increments of two. For each of these document sets we determine two different orderings to compare: (1) the search order modified such that the positively labeled documents are at the top, and (2) the ordering that would appear in *DocTable*, based on moving positively labeled documents to the top and using LEGO to try to improve the ordering of the rest.

Note that because of the randomization involved in the process, we repeat the entire process of picking, labeling, learning, and ordering ten times in order to obtain a more representative sample. Since constraints can only be generated when there is at least one document marked as relevant, we discard all those runs where there are no relevant documents. All values shown for precision and precision-at-n are averages across all experiments conducted for a particular search query.

We visualize the results of these experiments two different ways. For each search query we consider (1) how the precision is affected by the number of results shown to the user, and (2) how well *DocTable* performed compared to pure labeling as a function of how many labels a person had to provide. In the results of Fig. 6 there are two category labels making up the rows. For each category we show two search queries, each with a pair of graphs. The left plots of these pairs show

precision versus the position in the search result order (table order) at which it was calculated. These plots are a common way of looking at precision, since the total number of results affecting the precision calculation is not reliable across datasets (or queries). Precision itself falls as the number of results shown increases because later results are less likely to be relevant to the query. By plotting the precision of the ordering in *DocTable* compared with that of pure labeling, we can evaluate the overall strength of the retrieval. The green dots represent where in the baseline search (without simulated user input) the positive matches against the label occur. This information gives context to the changes in the precision values moving to the right in the graph.

The right plot of each pair demonstrates how *DocTable* can help a user find relevant documents by showing the effect on precision-at-n of increasing the number of labels. This test is closest to evaluating *DocTable* under usage conditions because it aims to show how LEGO magnifies the improvement gained from user effort. Specifically, we plot the precision values at the tenth row of the table of each of the scenarios: using user labels only (i.e., moving relevant, labeled documents to the top), and using *DocTable*'s model building functionality to augment labeling efforts.

8.1.2 Results

Although we ran this experimental process across a wide range of document labels and query combinations, we show the results for two selected categories of documents, "veg-oil" and "money-fx". Both of these categories demonstrate a case where the label was not a simple word we would expect in all relevant documents. Additionally, we chose labels with a relatively small number of relevant documents so that there would be some challenge in finding them. The plots in Fig. 6 show the results for these experiments. We have selected two queries to show in Fig. 6 for each category. For each of these categories, we evaluated multiple queries, as detailed above. The resulting plots were generally comparable and the results positive.

In plots (a), (c), (e), and (g) we see that the precision line for the scenario where LEGO is used shows a higher value, especially with moderate values of the x axis. This suggests that *DocTable* is able to create a better ordering of the documents. All x axes start at one because of the constraint that we have at least one positive label in our test set.

In plots, (b), (d), (f), and (h) we see a direct comparison between the *DocTable* and raw user labelling. The x axis represents how much work is involved for the user, i.e., how many documents they have marked. We can then see how *DocTable* is able to get more relevant results into the top of the table for a given amount of user effort.

Overall, this evaluation shows that the *DocTable* approach can help to impose a better ordering on a document set retrieved by search. Though the results displayed represent the typical behavior in our experiments, there are two cases we identify when applying our technique does not help. First, as is the case with any machine learner, is when the pattern to be learned cannot be represented by the type of model being used. LEGO is a well-vetted algorithm, but some themes will not be represented well by a linear model of the terms. For example, LEGO can capture when two documents share frequent usage of a set of words, but not if they were written at similar times. Relationships like being written by different people are not explicitly possible, but may be captured by differences in vocabulary between writers. Another situation in which *DocTable* performs less well in our evaluation is when the original search produces high quality results, i.e., relevant documents are plentiful and sorted to the top of the list. There is not much room for improvement, so LEGO may have little effect. It is possible it will worsen ordering, possibly due to lack of negative labels. When used in an interactive setting, this may not be a problem because it means the theme being investigated was easily represented by a search.

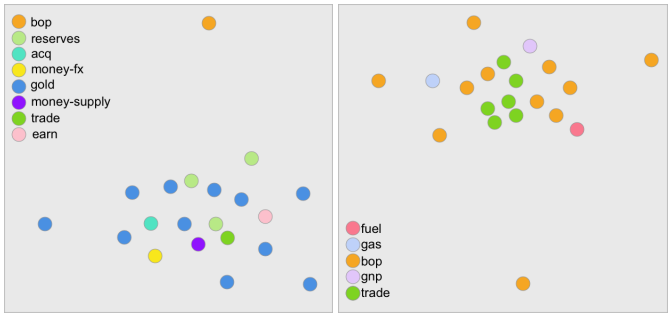
8.2 Evaluation of Model Influence on Visualizations

To evaluate the model influence on the term and document force-directed visualizations we used a similar experimental setup as above, again using the Reuters dataset. We chose a theme corresponding to a category label in the data, picked a related term to use as a search word, and then simulated interactions with the search results. Specifically, we choose documents randomly and supposed the user would label them based on the category label. Following the process in *DocTable*, the labels are used to create constraints and the constraints are used to learn a metric model with LEGO. This model is used to generate the force-directed graphs. Since the model reflects the simulated interactions, we can evaluate the capability of our method to reflect changes to the model in the visualization by comparing pre- and post-model versions by visual inspection.

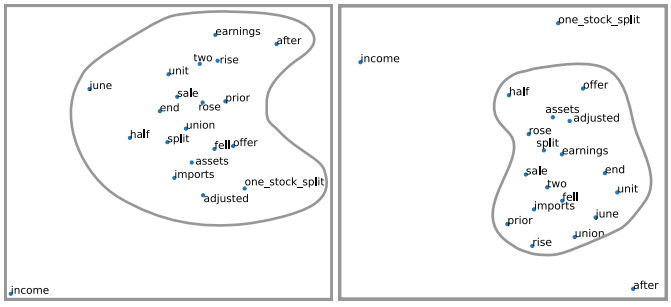
We chose the category label "trade" from the data and used the related term "deficit" as a search term. The 50 most relevant documents from the search were picked for interaction in the table. Ten were randomly chosen to label and marked as either "relevant" (if they had label "trade") or "non-relevant" (otherwise).

Fig. 7 shows the force-directed layouts generated using the Fruchterman Reingold algorithm [30] as implemented in the Python NetworkX package [33]. In the document visualizations, we show the 20 most significant documents and replaced the document names with their labels. This way, looking from (a) to (b) in the figure, we can see that applying labels has quickly shifted the emphasis so that we are seeing documents labeled 'trade' in the center of the group.

In the case of the terms force-directed visualization, Fig. 7 (c) and (d) show the affect of applying labels in the same manner. Although there is not much change in the terms themselves, we do see that their relative positioning has changed. For example, certain terms such as "imports" and "adjusted" have converged, indicating that there is a stronger relation between the two words based on interaction with *DocTable*. Further, the two graphs show that the term "income" is consistently detached from the main word cluster, indicating that it is likely unaffected by the model. In addition, the model has also increased the number of the terms pushed away from the main clustering. Given shifts to the layout in *DocTable*, the user would then look at documents and investigate relationships, hoping that the updated model helps home in on useful topics and content.



(a) Projection of Documents with No Metric (b) Projection of Document with Metric



(c) Projection of Terms with No Metric (d) Projection of Terms with Metric

Fig. 7: This figure shows examples of force-directed visualizations from our experiments to systematically verify the model was affecting the output. Outlines in (c) and (d) help emphasize the changing groups. See Sect. 8.2

.

9 Discussion and Future Work

The *DocTable* prototype was a helpful prototype to assess the utility of our approach to using interactive machine learning to help work through large text corpora. There is no shortage of interesting future work to pursue. Aside from interface features, there are many questions about the core interactive machine learning technology we can explore. For one, there are unexplored opportunities to learn from interactions, e.g., using interactions with the visualizations. Another opportunity is to switch our dimension reduction of the text data from PCA to Latent Dirichlet Allocation (LDA) [9], which would give us a similar vector space, but tailored for the problem of topic extraction in text.

From our understanding of the journalists' needs, they would be interested in tools to help understand the workings of the models better. They are not necessarily interested in the mechanics of the machine learning, but how to interpret what each model represents. Tools to describe and edit models could go a long way to assist comprehension and therefore trust. Further, this understanding will be important if people want to share models. For example, developing models that are useful for understanding policing based on data from Texas would not be helpful to an analysis in Washington state if one of the main salient features was the term 'Houston' (this is not a real example). A model might get good results on the Texas data by taking advantage of the name of a struggling city, but to generalize, the user needs trust and understanding.

10 Conclusion

In this paper, we presented interactive machine learning mechanisms for text data and a prototype demonstrating them, *DocTable*. We described how we designed *DocTable* with inspiration from our collaborator in journalism and used a usage scenario to show how it can be used to quickly arrive at interesting information hidden in large data. We also validated the technique with domain-expert feedback on the prototype from our collaborators. Our final evaluation was a series of simulation experiments to test the machine learning mechanisms to conclude they can use small amounts of feedback in our prescribed manner to improve models about concepts in text data. The techniques presented in this work can be applied to model learning and visualization for text corpora in a broad array of application areas.

References

[1] Agendaless Consulting and Contributors. Supervisor, 2018. Retrieved 17 SEP 2018 from http://supervisord.org/.

[2] Amazon Developers. EC2 documentation, 2018. Retrieved 17 SEP 2018 from https://aws.amazon.com/documentation/ec2/?id=docs_gateway.

[3] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 21–30. ACM, 2012.

[4] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian. Human-guided machine learning for fast and accurate network alarm triage. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pp. 2564–2569, 2011.

[5] C. Andrews and C. North. Analyst's workspace: An embodied sensemaking environment for large, high-resolution displays. In *IEEE Symposium on Visual Analytics Science and Technology*, 2012.

[6] S. Basu, S. M. Drucker, and H. Lu. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 394–400, 2010.

[7] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4), Nov 2016. doi: 10.1007/s00799-015-0156-0

[8] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[10] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[11] L. Bradel, C. North, and L. House. Multi-model semantic interaction for text analytics. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pp. 163–172. IEEE, 2014.

[12] Brainspace Corporation. Brainspace, 2017. Retrieved 17 SEP 2018 from https://www.brainspace.com/.

[13] E. T. Brown. *Learning from Users' Interactions with Visual Analytics Systems*. PhD thesis, Tufts University, 2015.

[14] E. T. Brown, A. Endert, and R. Chang. Human-machine-learner interaction: The best of both worlds. In *Workshop on Human Centered Machine Learning (HCML) at ACM CHI*, 2016.

[15] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-Function: Learning distance functions interactively. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 83–92. IEEE, 2012.

[16] E. T. Brown, S. Yarlagadda, K. A. Cook, R. Chang, and A. Endert. Modelspace: Visualizing the trails of data models in visual analytics systems. In *Proceedings of the Machine Learning from User Interaction for Visualization and Analytics Workshop at IEEE VIS*, 2018.

[17] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 539–546, 2005.

[18] CS Disco Inc. Disco, 2018. Retrieved 17 SEP 2018 from https://www.csdisco.com/.

[19] D3.js. d3-interpolate, 2018. Retrieved 17 SEP 2018 from http://github.com/d3/d3-interpolate.

[20] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[21] DocumentCloud. DocumentCloud main page, 2018. Retrieved 16 SEP 2018 from https://www.documentcloud.org/).

[22] A. Endert, R. Burtner, N. Cramer, R. Perko, S. Hampton, and K. Cook. Typograph: Multiscale spatial exploration of text documents. In *Proceedings of the IEEE International Conference on Big Data*, pp. 17–24. IEEE, 2013.

[23] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 473–482. ACM, 2012.

[24] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North. Observation-level interaction with statistical models for visual analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 121–130. IEEE, 2011.

[25] A. Endert, W. Ribarsky, C. Turkay, B. Wong, I. Nabney, I. Díaz Blanco, and F. Rossi. The state of the art in integrating machine learning into visual analytics. *Computer Graphics Forum*, 3 2017.

[26] Everlaw. Everlaw main page, 2018. Retrieved 17 SEP 2018 from https://www.everlaw.com.

[27] Facebook. React library page, 2018. Retrieved 17 SEP 2018 from https://github.com/facebook/react.

[28] Flask contributors. Flask (python web microframework), 2017. Retrieved 31 MAR 2018 from http://flask.pocoo.org/.

[29] J. Fogarty, D. Tan, A. Kapoor, and S. Winder. Cueflik: interactive concept learning in image search. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 29–38. ACM, 2008.

[30] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[31] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of the International Conference on Computer Vision*, pp. 498–505, 2009.

[32] Gunicorn Developers. Gunicorn documentation, 2018. Retrieved 17 SEP 2018 from https://github.com/benoitc/gunicorn.

[33] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab (LANL), Los Alamos, NM (United States), 2008. networkX python library.

[34] C. Hagerman, R. Brath, and S. Langevin. Visual analytic system for subject matter expert document tagging using information retrieval and semi-supervised machine learning. In *Proceedings of the 23rd International Conference Information Visualisation (IV)*, pp. 234–240, 2019. doi: 10.1109/IV.2019.00047

[35] S. Havre, B. Hetzler, and L. Nowell. Themeriver: Visualizing theme changes over time. In *IEEE Symposium on Information Visualization (InfoVis)*, pp. 115–123. IEEE, 2000.

[36] F. Heimerl, S. Koch, H. Bosch, and T. Ertl. Visual classifier training for

text document retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2839–2848, 2012. doi: 10.1109/TVCG.2012.277

[37] S. Hoi, W. Liu, M. Lyu, and W. Ma. Learning distance metrics with contextual constraints for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2072–2078, 2006.

[38] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pp. 761–768, 2009.

[39] D. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An interactive system for PCA-based visual analytics. *Computer Graphics Forum*, pp. 767–774, 2009.

[40] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[41] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. Retreived 28 SEP 2021 from http://www.scipy.org.

[42] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. *Visual analytics: Definition, process, and challenges*. Springer, 2008.

[43] S. Leman, L. House, D. Maiti, A. Endert, and C. North. Visual to parametric interaction (v2pi). *PLoS ONE*, 8(3):e50474, 2013.

[44] LexisNexis. Ravel law, 2018. Retrieved 17 SEP 2018 from https://home.ravellaw.com/.

[45] Logikcull. Logikcull, 2018. Retrieved 17 SEP 2018 from https://logikcull.com/.

[46] B. McFee and G. R. Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 775–782, 2010.

[47] T. Muhlbacher and H. Piringer. A partition-based framework for building and validating regression models. *Transactions on Visualization and Computer Graphics*, 19(12):1962–1971, 2013.

[48] NGINX. Nginx main page, 2018. Retrieved 17 SEP 2018 from https://www.nginx.com/.

[49] T. E. Oliphant. *A guide to NumPy*. Trelgol Publishing, 2006.

[50] Overview. Overview main page, 2018. Retrieved 16 SEP 2018 from https://www.overviewdocs.com/.

[51] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50. ELRA, Valletta, Malta, May 2010. `http://is.muni.cz/publication/884893/en`.

[52] D. Sacha, M. Sedlmair, L. Zhang, J. Lee, D. Weiskopf, S. North, and D. A. Keim. Human-centered machine learning through interactive visualization: review and open challenges. In *In Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2016.

[53] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012. doi: 10.1109/TVCG.2012.213

[54] J. Z. Self, M. Dowling, J. Wenskovitch, I. Crandell, M. Wang, L. House, S. Leman, and C. North. Observation-level and parametric interaction for high-dimensional data analysis. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–36, 2018.

[55] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[56] J. Stasko, C. Görg, and Z. Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.

[57] The Jquery Team. jQuery documentation, 2018. Retrieved 17 SEP 2018 from https://github.com/jquery/jquery.

[58] G. Trivedi, P. Pham, W. W. Chapman, R. Hwa, J. Wiebe, and H. Hochheiser. Nlpreviz: an interactive tool for natural language processing on clinical text. *Journal of the American Medical Informatics Association*, 25(1):81–87, 2017.

[59] S. van den Elzen and J. J. van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pp. 151–160. IEEE, 2011.

[60] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. Brown, and A. Endert. Podium: Ranking data using mixed-initiative visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[61] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang. Tiara: a visual exploratory text analytic system. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 153–162, 2010.

[62] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 19*, pp. 10:207–244, 2006.

[63] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Information Visualization, 1995. Proceedings.*, pp. 51–58. IEEE, 1995.

[64] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pp. 505–512, 2002.

[65] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, pp. 1–51, 2006.