Article

# TaskFinder: A Semantics-Based Methodology for Visualization Task Recommendation

**Darius Coelho [1,†], Bhavya Ghai [1], Arjun Krishna [1], Maria Velez-Rojas [2], Steve Greenspan [2], Serge Mankovski [2], and Klaus Mueller [1,*]**

[1] Department of Computer Science at Stony Brook University 1
[2] CA Technologies (at the time of this work)
* Correspondence: dcoelho@cs.stonybrook.edu
† Current address: Stony Brook University, Stony Brook, NY 11790.

**Abstract:** Data visualization has entered the mainstream and numerous visualization recommender systems have been proposed to assist visualization novices, as well as busy professionals, in selecting the most appropriate type of chart for their data. Given a dataset and a set of user-defined analytical tasks, these systems can make recommendations based on expert coded visualization design principles or empirical models. However, the need to identify the pertinent analytical tasks beforehand still exists and often requires domain expertise. In this work we aim to automate this step with TaskFinder, a prototype system that leverages the information available in textual documents to understand domain-specific relations between attributes and tasks. TaskFinder employs word vectors as well as a custom dependency parser along with an expert-defined list of task keywords to extract and rank associations between tasks and attributes. It pairs these associations with a statistical analysis of the dataset to filter out tasks irrelevant given the data. TaskFinder ultimately produces a ranked list of attribute-task pairs. We show that the number of domain articles needed to converge to a recommendation consensus is bounded for our approach. We demonstrate our TaskFinder over multiple domains with varying article types and quantities.

**Keywords:** Visualization Recommendation; Natural Language Processing; Visualization Systems and Tools

**Citation:** Coelho, D.; Bhavya, G.; Krishna, A.; Velez-Rojas, M.; Greenspan, S.; Mankovski, S.; Mueller, K.;. TaskFinder. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:



## 1. Introduction

The recent exponential increase in data generation activities has pushed data visualization into the mainstream. However, many people lack the expertise or resources to generate insightful data visualizations. To address this issue, researchers have proposed various visualization recommender systems. These systems generally function by taking a dataset as input, supplemented by any necessary additional input from the user, and generating a ranked list of recommended visualizations as output.

The early visualization recommender systems [1,2] focused on suggesting a list of visualizations based on design criteria such as *effectiveness* and *expressiveness*. Subsequent iterations integrated statistical properties of the data into their recommendations. Later, researchers demonstrated the influence of visualization types on a user's task-based performance [3]. This led to more recent systems requiring users to specify their intended *low-level analytic tasks* [4] before receiving visualization suggestions [5]. While it is reasonable to require users to select analytic tasks, their inexperience or the size and complexity of their data may cause them to overlook important tasks.

In this work, we propose a methodology that leverages information within textual documents to determine the most relevant attributes in a tabular dataset and the corresponding analytic tasks. While many recommender systems, especially in the visualization field, rely on expert-crafted rule-based algorithms or models derived from empirical data, these might not be optimal for suggesting visualization tasks. Attributes of interest and recommended tasks can greatly vary across different data domains thus making it prohibitively expensive to design rules or collect model training data through empirical studies. Instead, we show that it is possible to use a previously untapped source of information - textual documents - to serve this purpose.

People or organizations often post data-driven articles that explain, review, and make comparisons on a variety of topics on the web. We conduct a preliminary investigation that demonstrates such documents on the web do in fact report (directly or indirectly) the analytic tasks they performed on various attributes to make their findings. Thus we believe that extracting this information will allow us to drive a visualization task recommender system. To this end, we develop a technique that leverages a combination of natural language processing (NLP) techniques (e.g. Part-of-speech or POS tagging, dependency parsing, semantic word embeddings) to automatically identify data attributes and analytic tasks in natural language (NL) documents such as magazines and reviews.

To demonstrate the application of this extraction method to a visualization recommender system, we introduce a prototype system called TaskFinder. Operating on a user-provided tabular dataset and a set of relevant text documents, TaskFinder extracts attributes and mentions of analytic task. Next, it determines the importance of data attributes and associated analytic tasks based on textual frequency, order of appearance in documents, and statistical analysis of the dataset. This information is then utilized to propose suitable visualizations for investigating the data, ranked by importance. The recommended list of visualizations is provided to the user via the interface shown in Figure 4.

We demonstrate TaskFinder across two distinct domains, showcasing how it delivers recommendations. We show that both the quantity and quality of texts impact the recommendations, highlighting the significance of document selection. Additionally, we reveal that as the quantity of texts grows, the recommendations tend to converge toward a consensus.

In summary, our contributions are as follows:

- A preliminary study demonstrating the viability of leveraging web-based textual information to associate visualization tasks with dataset attributes.
- A method that leverages NLP techniques to extract attribute importance and associated analytical tasks from textual documents.
- A prototype system - TaskFinder - which implements a ranking method that combines information gained from the NL attribute-task extraction method and the statistical properties of a dataset to recommended visualizations.
- A demonstration of TaskFinder with two data domains.
- An evaluation showing that the number of documents needed to converge to a recommendation consensus is bounded.

## 2. Related Work

### *2.1. Visualization Recommenders*

Visualization recommender systems aim to lower the barrier for data analysts who lack the expertise or time required to visually represent their data. In most cases, these systems start off by asking users to specify the dimensions or data attributes they are interested in and the task they wish to perform. Once this is determined a rule-based or machine-learning approach is typically employed to filter and rank the appropriate visualizations.

APT[1], a system developed by Mackinlay was one of the first attempts at building a visualization recommender. It is a rule-based system that uses composition algebra and design criteria based on works by Bertin [6] and Cleveland et al. [7] to suggest effective graphical presentation designs. Later, the SAGE [2] and BOZ [3] systems built upon APT, also considering the statistical properties of the data as well as the analytical tasks a user wishes to perform. AutoBrief [8] and AutoVis [9] further extended these works by supporting additional visualizations and statistical analyses. Most recently, SeeDB [10] provides recommendations of aggregate views using similar statistics while improving computational performance.

The systems discussed above require users to specify data attributes and the analytical task they wish to perform on them. However, some users may not have any tasks in mind; this is typical of users inexperienced in data analysis. Mackinlay et al. sought to address this issue with Tableau's Show Me [11] interface commands. Show Me supports the user's search for visualizations by suggesting good defaults for the visualizations using heuristic rules. Key et al. [12] also aimed to help users build task-appropriate dashboards with VizDeck by letting them select visualizations from a ranked list of visualization thumbnails. VizDeck uses the statistical properties and user voting to learn a scoring function that is used to rank visualizations. More recently the interactive systems Voyager [13] and Voyager2 [14] allow users to navigate a gallery of recommended visualizations. A unique feature of their approach is that as the user selects visualizations or attributes the gallery is updated. A recent effort is the work by Lee et al. [15] who offer analysts several paths, such as enhance, generalize, and pivot by which they can transition from one visualization to another.

Recently, multiple machine learning-based approaches have been proposed to recommend visualizations. Saket et al. [5] evaluated the effectiveness of a set of visualizations across ten visualization tasks put forth by Amar et al.[4]. The findings of this study were then used to train a decision tree for a visualization recommender they called Kopol. Lou et al. [16] also used data from an empirical study to train a decision tree that decides if a visualization is good or bad and a learning-to-rank model to rank the visualizations. The Draco [17] system models visualization design knowledge from empirical studies as a collection of constraints, it also uses a learning-to-rank model to train its recommender engine and easily allows its knowledge base to evolve with newer studies. Data2Vis [18] is an end-to-end visualization generation system. Given a dataset, it provides a valid Vega-Lite specification. It applies a neural machine translation (seq2seq) model that is trained with a large number of datasets and their visualizations in Vega-Lite specification to learn appropriate transformations (count, bins, mean) and common data selection patterns. VisML [19] developed a recommendation strategy by learning the association between dataset features and visualization properties. KG4Vis [20] uses a knowledge graph to encode these associations which adds transparency to this process. Finally, MultiVision [21] extends the recommendations to construct entire dashboards by adding a set of guidelines. These systems are all able to predict the visualization type or how data should be visually encoded however they do not predict visualization tasks.

These prior systems made significant strides toward visualization recommendation. However, they still require some effort on the user's part to determine attributes of interest and the analytic tasks to be performed. Additionally, they do not explicitly consider the data domain, which is a factor that can significantly affect the choice of task and visualization. TaskFinder aims to augment these systems with a novel method that recommends attributes and analytical tasks based on the specific domain of the data.

### 2.2. NLP in Visualization

Recently, NLP techniques have matured and are being applied in many domains. One such technique that is widely applied in the field of visualization is word embeddings. Word embeddings are vector representations of words in a high-dimensional geometric space often referred to as *vector space*. Multiple models have been proposed to learn these embeddings from a large corpus of text. These models use the context of a word, i.e. its surrounding text, to map it to the vector space. Thus, words that share the same context appear closer to each other in vector space and are said to be semantically similar or related. Berger et al. [22] extended the Word2Vec embedding technique to embed both words and documents in the vector space and then visualized this space to show the relationship between documents and words. Park et al. [23] created ConceptVector, a visual analytics system that helps users refine concepts generated from word embeddings and use these concepts to analyze documents. Mahmood et al. [24] used data analytics paired with word embeddings of data attributes and their context to help users build taxonomies. Our work employs word embeddings as well; we use them to determine if words in textual documents refer to attributes and tasks.

In addition to using NLP techniques to analyze texts for visualization purposes, researchers in the visualization field have recently been developing natural language interfaces (NLI) for visualization. Cox et al. [25] created one of the first NLIs for visualization. They use a grammar-based approach to convert NL questions into database queries and return the results to the user via tables and bar charts. Sun et al. [26] followed up on this work with their Articulate system that considers various low-level analytical tasks and returns a wider range of visualizations. DataTone [27] allows users to specify visualizations through NL queries while detecting ambiguities in those queries using a combination of lexical, constituency, and dependency parsing. Setlur et al. [**?** ] developed Eviza which implements a probabilistic grammar-based approach and a finite state machine to allow people to interact with a given visualization using NL commands. Flowsense [28] employs a semantic parser to parse NL queries that manipulate multi-view visualizations produced by a dataflow diagram. The system allows users to expand and adjust dataflow diagrams more conveniently. Most recently, Narechania et al. identified the popularity of NLI in visualization and developed the NL4DV toolkit [29] to aid visualization developers who may not have a background in NLP develop NLIs for visualization.

TaskFinder is closely related to these NLI. It attempts to extract attributes and associated task mentions from large amounts of text in articles or reviews. This is akin to the way NLIs understand users' NL queries However, the queries these NLI systems process are more direct and tend to be less ambiguous. For example, users specifically ask the system to "show the correlation between horsepower and MPG" which the system understands as applying the *correlation* task to the *horsepower* and *MPG* data attributes. TaskFinder, on the other hand, deals with articles or reviews that essentially report the result of an analysis and it must infer which task and attributes were used to generate the result. Consider, for example, the sentence “The Hyundai's powerful engine leads to a lower fuel economy". Here, it can be inferred that the words *powerful* and *fuel-economy* are referring to the *horsepower* and *MPG* data attributes and the phrase *leads to* implies that the *correlation* task was used to deduce this relationship. To make such inferences, we expand on the approaches discussed above.

## 3. Preliminary Study: Can we learn from text on the web?

The internet is a rich source of textual information; however, a large portion of it is irrelevant to our task of learning how people use visualization for analysis tasks. Before devising a method to analyze texts for this purpose, we needed to assess whether online documents contain relevant content that links tasks to attributes. Our initial exploration is outlined below.

When exploring data, people usually ask questions about the data and we hypothesized that we could uncover task-attribute connections from such queries. For instance, an individual might ask, “what is the maximum horsepower of all cars?”. Here, we observe that the task *find extremum* is represented by the word ‘maximum' which is applied to the attribute ‘horsepower'. Initially, we speculated that rather than conducting a structured investigation to formulate such questions, we could extract them from the Internet. Typically, such queries are posted on question-answer forums like Question.com or within the "People Also Asked" section of Google's search results. Hence, we conducted a preliminary study to determine the feasibility of systematically locating these questions and extracting task-attribute relationships.

To start off, we selected two datasets for this study - cars [30], and NBA player data [31]. These datasets were specifically chosen as they appeal to a large audience which implied that a large number of questions would be available. We found that querying the forums with the dataset name or topic along with one or more attributes returned a list of questions related to those attributes. A large portion of these questions could be answered by conducting analytical tasks over the datasets. For example, a search for the term "Cars weight acceleration" returned the question "How does weight affect acceleration?" which can be answered by investigating the *correlation* between the "weight" and "acceleration" attributes in the dataset. The tasks we consider here are the ten low-level analytic tasks put forth by Amar et al. [4] which are used in multiple prior visualization recommenders. These tasks are - *Find Anomalies*, *Find Clusters*, *Find Correlation*, *Characterize Distribution*, *Determine Range*, *Find Extremum*, *Order*, *Filter*, *Compute Derived Value*, and *Retrieve Value*.

The results of our preliminary exploration encouraged us to build a collection of questions that we could study and use to learn about task-attribute relationships in the questions. We consequently collected 342 questions for our datasets. Next, two authors independently coded these questions by identifying the task and associated attributes in a question. As questions tend to be highly specific, each question was assigned a single task and one or more attributes. Each coder worked independently and jointly resolved any disagreements on attributes or tasks through discussions. After the coding process, we were left with 255 questions that referred to an attribute and associated tasks; the remaining questions were invalid. Invalid questions primarily focused on conceptual queries or explanations of terminologies, such as questions about car components or basketball jargon. These questions, like "What does horsepower mean?" or "What is a wing position in basketball?", were not related to the focus of our work. Although they might aid users in understanding the underlying concepts, they are not easily answered through data analysis.

Analyzing the corpus of questions, we found that a majority (over 75%) of the questions revolved around *finding extremums* or *retrieving values* whereas fewer (just over 20%) questions were categorized as *finding clusters* or *characterizing a distribution* and none were related to *finding anomalies*. The results are shown as a percentage in Figure 1 (blue bars). These questions tend to be posted by the general public who are non-experts or people looking for elementary information and hence lack analytical depth. Furthermore, a comparison with the types of questions a data analyst or visualization expert might pose revealed the relatively straightforward nature of these queries; they mostly involved univariate analyses.

These findings prompted us to delve deeper, querying domain experts rather than the general public to unearth questions akin to those posed by proficient analysts. But this

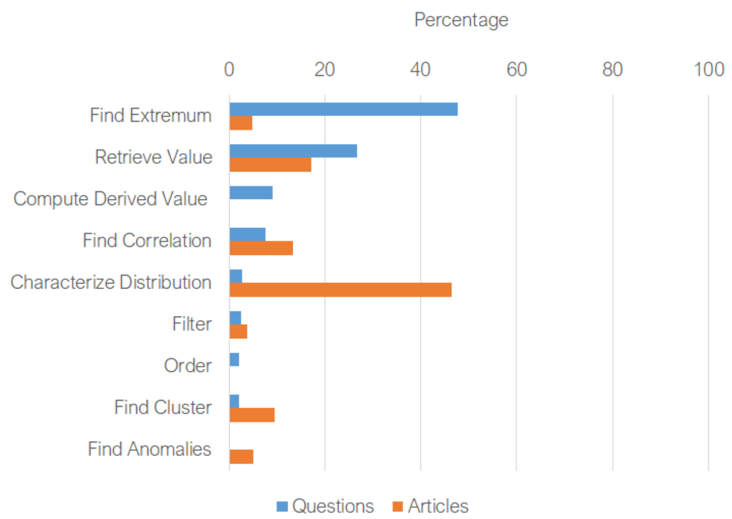



**Figure 1.** Here we show the results of our preliminary analysis of NL texts on the web. We report the percentage of questions (from forums) and the percentage of sentences (from articles) that are associated with the 10 low-level analytical tasks. We observe that articles tend to be more detailed and focus on some of the more involved tasks such as finding clusters and anomalies. In both cases, the *Determine Range* task was not present.

does not mean that we arrived back at square one, i.e., conduct formal interviews with domain experts to gain this knowledge. Rather, we could harness a wide array of possibly large domain text available on the web.

Upon further exploration, we observed that data-driven news articles, review articles, and enthusiast blogs posted on the internet contained detailed information gained from data analysis. Authored primarily by domain experts or enthusiasts, these articles are inherently rich in information and analytical in nature. For example, when analyzing the car dataset, we could leverage comparison reviews from car magazines as the writers are knowledgeable about the domain (cars) and they compare all aspects (attributes) of the cars. To confirm the utility of such articles, we collected six articles related to the same datasets - three car review articles and three NBA player profile articles. The sentences in each article were coded following the same procedure used for the questions. The results are also shown as a percentage in Figure 1 (orange bars). The results indicate that these articles do in fact tend to refer to some of the more analytical tasks such as *finding clusters* (approximately 10% of the task mentions) or *finding anomalies* (approximately 5% of the task mentions) as compared to the questions found on forums. We also considered reviews posted by consumers but encountered similar issues of naivety as with the questions we had gathered. These findings encouraged us to develop an automated technique for analyzing dataset-related articles, enabling the extraction of insights that inform us of attribute significance and associated analytical tasks.

## 4. Design Requirements

The overall goal of our work is to recommend a set of visualizations that allow users to explore important features in a tabular dataset informed by data attributes' importance and associated analytic tasks. Fundamental to our approach is the belief that analysis-type articles related to the subject of the dataset can be useful in providing us with information relating to an attribute's importance and the analytic tasks associated with it.As mentioned, this led us to build *TaskFinder*, a prototype system that leverages a combination of NLP techniques and statistical methods to extract information from a dataset and related textual documents and generate a list of recommended visualizations based on the importance of data attributes and associated tasks. Based on observations made in our preliminary study, we identified four main design requirements for TaskFinder:

**R1** *Identify all mentions of attributes in the article texts*. First, TaskFinder should identify all occurrences of words referencing a dataset attribute in the article text. This is not a straightforward process as people tend to use different words to refer to the same attribute (e.g. 'mpg' and 'mileage' can be used interchangeably). Also, a single word could refer to the same attribute and task together, for example, 'fastest' refers to the *find extremum* task applied to the 'acceleration' or 'speed' attribute. Thus TaskFinder must be able to identify all words referring to tasks and attributes.

**R2** *In each sentence identify tasks and determine their relationship to attributes*. After determining which words refer to attributes, the next requirement of our system is to determine if and how these words are associated with words referencing tasks in the same sentence. That is, determine which tasks are applied to attributes. At times, sentences refer to multiple tasks and attributes and our system must be able to identify which tasks are applied to which attributes. For example, in the sentence "During our longest drive the BMW gave us an average of 34.2 mpg.", there are two tasks - the *find extremum* task referred to by the word 'longest' which is applied to the word 'drive' and the *compute derived value* task referred to by the word 'average' which is applied to the word 'mpg'. TaskFinder must be capable of making this distinction.

**R3** *Compute importance and rank*. Completing the tasks above would result in multiple attribute-task relationships being identified. Depending upon the number of attributes and tasks mentioned, this list could be extremely long. Thus our system's third requirement is to rank the task attribute pairs and provide the user with the most important pairs first.

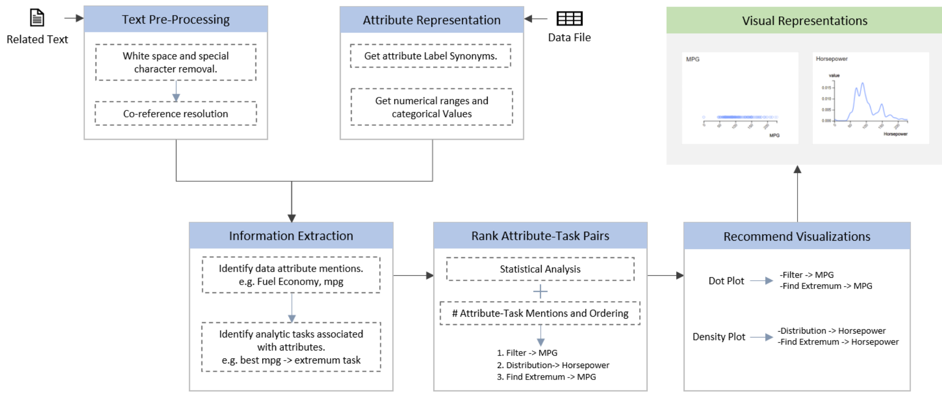



**Figure 2.** The workflow of TaskFinder. First, the user provides TaskFinder with tabular data and a set of domain-related documents. Next, the documents are cleaned with text pre-processing methods while TaskFinder creates a representation for attributes in the data. Next, this representation is used to extract information from the documents and create an association between attribute mentions and tasks associated with them. These attribute-task pairs are then ranked by frequency of appearance in documents as well as statistical properties of the data. Finally, a visualization is recommended for each attribute-task pair and is provided to the user as a list of visualizations.

**R4** *Recommend appropriate visualizations for each attribute-task pair*. Our main goal is to provide users with appropriate visualizations based on the type of data and analytical task to be performed. Thus our final requirement is to determine a mapping between visualization tasks, attribute types, and visualizations.

## 5. TaskFinder

An overview of TaskFinder's workflow is shown in Figure 2. The user starts by providing TaskFinder with a tabular dataset and a set of related textual documents which we will refer to as the *corpus*. It initially performs text pre-processing to clean the corpus. Next, it extracts information from the dataset such as attribute labels and properties such as range and categorical values to form an attribute representation. This representation is used to identify references to attributes in the corpus (**R1**). Sentences containing attribute mentions are analyzed to infer if one or more analytic tasks are associated with the attribute(s) (**R2**). The frequency and order of appearance of attributes and associated analytic task mentions are then used to generate a semantic ranking of the attributes or attribute pairs and associated tasks. We perform statistical tests on the dataset to rule out certain tasks and generate an interestingness score for each attribute or attribute pair. We then combine the statistical ranking and semantic ranking to generate a combined ranking (**R3**). Finally, based on the attribute types (numerical, nominal, or time) and associated tasks we recommend appropriate visualizations and provide them to the user via a web-based interface shown in Figure 4 (**R4**). Each of these processes are explained below.

### 5.1. *Attribute Representation*

Authors often use different words to refer to the same term or concept across text documents on the web. Thus, it is likely that an attribute in the dataset is referred to by multiple different terms or words in the textual documents. For example, the attribute 'MPG' in the cars dataset might be referred to as 'fuel economy' in the text. Additionally, categorical attributes might be referred to by their categories instead of the attribute name. For example, the word 'USA' may be used to refer to the attribute 'Origin' which reports the country of manufacture for a car in the dataset.

In order to address this issue, we must represent each attribute in the dataset by a collection of words instead of just the attribute label. We do this by representing each attribute by its label and a set of synonyms that we generate automatically. We make use of Datamuse [32] and NLTK [33] synsets to generate a list of attribute synonyms. We also add the categorical values of categorical attributes to the collection of synonyms that represent the attribute. It should be noted that we do not add synonyms of categorical values. In some cases, the synonyms are not applicable to the domain of the data thus we allow the user to interactively deselect the irrelevant synonyms via the interface shown in Figure 3. We limit the number of synonyms to 30. We found this to be sufficiently large when experimenting with a number of datasets retrieved from Kaggle. The user can edit attribute types and attribute names which would lead to a new refined subset of synonyms.

### *5.2. Pre-processing*

Text documents on the web do not conform to any particular standard or format and may contain special characters and white spaces that can affect the performance of many NLP tools. Thus, like many natural language processing systems, we must pre-process the corpus before we actually analyze it. First, we remove any accented characters and extra white spaces. Next, we perform coreference resolution which is the task of finding all expressions that refer to the same entity across a set of sentences. We used Spacy's [**?** ] implementation of the coreference resolution published by Clark and Manning [34] For example, performing coreference resolution "The car's fuel efficiency is 24.3mpg. It is the best in its class." replaces the first instance of "It" in the second sentence with "The car's fuel efficiency" and the second instance of "It" is replaced with "The car". This helps our system to detect that the *extremum* task ("best") was applied to the fuel efficiency attribute in the second sentence. Once we have pre-processed the corpus we move on to analyzing it sentence by sentence.

### *5.3. Information Extraction*

To recommend important attributes and their related tasks, TaskFinder must extract useful information about them from the corpus. To achieve this, we make use of a combination of NLP techniques, specifically part-of-speech (POS) tagging, named entity recognition (NER), dependency parsing, and word embeddings. These techniques are implemented by a variety of NLP toolkits; for our work we make use of NLTK [33], Spacy [**?** ], and Gensim [35] with ConceptNet's [36] word vectors. We discuss how we use these techniques to identify references to data attributes in a sentence and how we infer tasks applied to the attribute. Our discussion includes terminology common in the field of NLP, for a brief description of these terms please refer to the NLP dictionary created by Wilson [37].

#### 5.3.1. Parsing Sentences

We start by iterating over each sentence in the pre-processed corpus and using Spacy, we apply a series of NLP functions to them to extract features that can be used to detect attributes and associated analytic tasks. We first perform POS tagging and extract the POS tag (e.g. *NN*: Noun, *JJ*: Adjective, *VB*: Verb etc.) of each token in the sentence. Next, we perform NER and extract all named entity tags (e.g. *GPE*: Countries, cities, states, *PERSON*:

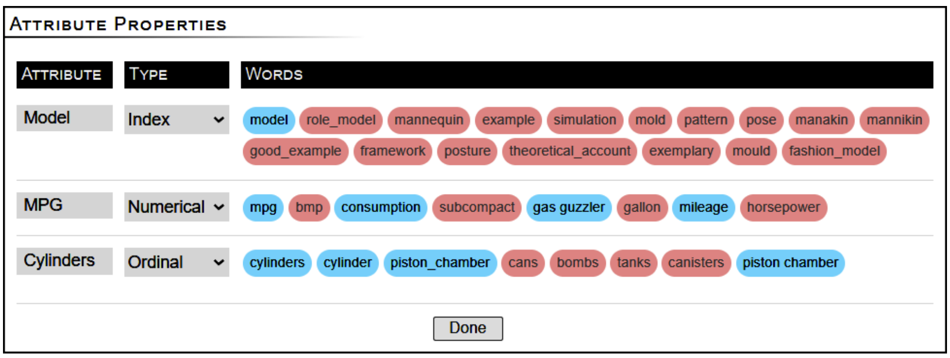



**Figure 3.** TaskFinder's interface allows the user to guide the attribute representation. Here users can set the attribute types, edit attribute labels, and deselect irrelevant synonyms.

names of people, *PRODUCT*: names of products, *QUANTITY*, *TIME* etc.) for words or phrases in the sentence. Now, to understand the connection between words and phrases in the sentence, we extract dependency relations between words as a dependency tree using Spacy's dependency parser. Finally, to identify certain phrases (e.g. "greater than", "leads to") we construct N-grams (a collection of $N$ successive items) from the sentence tokens.

#### 5.3.2. Identifying Attribute Mentions

Once we have parsed the corpus, our first objective is to identify the data attribute mentions. As discussed above, multiple different words in the corpus may refer to the same attribute. For example nouns or noun phrases like 'fuel economy' or 'fuel efficiency' both refer to the 'MPG' attribute. Additionally, the corpus may contain adverbs and adjectives that may refer to attributes. For example, the words 'fastest' and 'quickly' refer to the 'acceleration' attribute. To identify these types of attribute mentions we make use of word embeddings [38] [39] [40], POS (part-of-speech) tagging, and named entity recognition.

To identify attributes, we only consider N-grams with the noun, adjective, verb, and adverb POS tags along with all tokens tagged as named entities in the sentence. Based on the properties of word embeddings we expect that words related to a data attribute appear closer together in the vector space. Thus, to determine if an N-gram or named entity refers to an attribute we compute the semantic similarity score between every N-gram and synonym and categorical value in the data attribute representation using their word vectors. If the semantic similarity (absolute value between 0 and 1) is very high and above a preset threshold, we count that word as a mention of the attribute. For nouns, we found that a threshold of 0.45 has worked well to filter out irrelevant words. For adjectives, verbs, and adverbs a lower threshold of 0.35 worked well as these words are more loosely connected to attributes in the word embedding space. If a word is marked as an entity, we follow a different strategy. For entities with the tag *ORG*, *NORP*, *GPE*, *PERSON*, *PRODUCT*, and *LANGUAGE* we only test for semantic similarity with attributes that are categorical. We also set a high threshold of 0.6 for the similarity. If an entity is tagged as the *DATE* or *TIME* we test for the semantic similarity between the tag label i.e. 'date' or 'time' and the synonyms of the attribute marked as time. As some unique words or phrases may not be present in ConceptNet's vocabulary, we also look for direct matches between N-grams and values, words, or phrases in the attribute representations and assign all matches to their respective attributes. It should be noted that we determined the threshold values by experimenting with a set of 15 car review articles, 10 NBA player profiles, and 7 data-driven news articles.

#### 5.3.3. Identifying Tasks Applied to Attributes

The next objective is to identify and associate analytical task with the identified attributes in the sentences. We analyze each sentence for mentions of any of the ten low-level tasks put forth by Amar et al. [4]. In this stage, we make use of word embeddings, POS tagging, named entity recognition, and dependency tree parsing.

These NLP techniques are used in conjunction with a set of keywords that refer to each task defined by us. We constructed the set task keywords by having a pair of experts (who are authors) refine a list of machine-generated synonyms. We generated the initial list of synonyms by first using a synonym generator, Datamuse [32], to generate a set of root synonyms. We then use these synonyms to retrieve the top 200 most related words to the synonyms based on their distance in the ConceptNet word embedding space. Using the number 200 proved to be sufficient to extract some closely related words as well as different forms of the same word. For example, the root word 'anomaly' has the words 'anomalies', 'anomalistic', 'anomalous', and 'anomalously' associated with it. Following this approach, we collected over 5,000 unique words and phrases that represent the ten tasks, with each task having 450 to 750 keywords each. These keywords were then filtered by two co-authors, with each author removing a word he or she found to misrepresent the task. Each author worked independently to remove words. The results were then merged

**Table 1.** This table lists the number of visualization guides that recommend using a chart for a particular task.

| Tasks | Charts | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Bar | Pie | Line | Density Plot | Box Plot | Dot Plot | Scatterplot | Parallel Coordinates | Stacked Bar | Balloon Plot | Heatmap |
| Distribution | 11 | 0 | 0 | 8 | 7 | 1 | 8 | 0 | 0 | 0 | 1 |
| Comparison | 18 | 16 | 13 | 0 | 1 | 0 | 7 | 3 | 7 | 2 | 7 |
| Relationship | 14 | 0 | 16 | 0 | 0 | 0 | 16 | 3 | 3 | 2 | 7 |
| Range | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

with conflicts (a word present in one author's list but not the other's) resolved through discussions. We then only used the words that were common to both author lists resulting in a total of 1,321 task keywords.

References to tasks occur in different manners in the text, thus we are using a different combination of NLP techniques to identify each task. First, we identify references to the *correlation*, *anomalies*, *cluster*, *derived value*, and *distribution* tasks following a procedure similar to that used for detecting attribute mentions. Here, we compute the semantic similarity score between every N-gram and attribute keyword using their word vectors. Words with a similarity score of more than 0.4 are considered references to tasks. Next, we make use of POS tagging to determine if a word is referring to the *extremum* and *range* tasks or *filter* and *rank*. We consider *extremum* and *range* (the two extremes) as essentially being the same task and group them into a single *extremum*. Similarly, we group the *filter* and *order* tasks into a single *filter* task as they both require to compare values. Then N-grams tagged as *JJS* or *RBS* i.e. superlatives (e.g. best, fastest, etc) and *JJR* or *RBR* i.e. comparatives (eg. bigger, faster, etc) are assigned the *extremum* and *filter*, respectively. We also look for direct matches between N-grams and task keywords as we did with the attributes.

Now that we identified all N-grams referring to data attributes and tasks, we must determine the association between the attributes and tasks. We parse the dependency tree using rules based on a combination of POS tags, dependency types(e.g. *nsubj*, *amod*), and tree distance to identify associations between tasks and attributes. The dependency parsing rules were defined based on the rules developed for NL4DV and the patterns observed in the $\sim$300 sentences analyzed in our preliminary study. Finally, if an attribute is mentioned in a sentence but none of the above tasks are associated with it, we default to recommending the *retrieve value* task if it is a named entity otherwise we recommend the *distribution* task. As a result of the process, we are left with attribute-task pairs such as *(Horsepower, Extremum)* for univariate analysis or *(MPG | Horsepower, Correlation)* for bivariate analysis.

### 5.4. Statistical Analysis

With the processes described above, we are able to extract attribute mentions along with tasks associated with them based on the textual documents provided. However, these documents are related to the domain of the data and not the dataset itself. Thus it may be the case that a task recommended based on the analysis of the text is not statistically interesting to perform. For example, the information extraction process may find that the clustering task is strongly associated with an attribute. However, the attribute may not have any clusters in the data. In this case, the strength of the association between the task and the attribute must be reduced.

We generate statistics for four tasks 'Anomalies', 'Clusters', 'Correlation', and 'Ordering'. We compute the number of outliers, clusters, sortedness (univariate only), and correlation coefficient (bivariate only) across all attributes. For nominal or ordinal attributes we compute these statistics over the counts of their values. While there are no statistics to rank attributes for other tasks, we compute an interestingness score for attributes based on general statistics - dispersion, variance, entropy, and skewness.

### *5.5. Ranking Attributes and Associated Tasks*

After parsing the user-provided text, we are left with a list of attribute-task pairs. This list can be very long if the user provides a large number of texts. Additionally, the texts are related to the domain of the dataset and not the dataset itself, thus the list may contain task-attribute pairs that may not be relevant to the dataset. Thus in its final step, TaskFinder must rank these task-attribute pairs based on some measure of importance. We compute this importance measure based on three metrics - pair frequency, pair sequence, and the statistical properties of the dataset.

- **Frequency ($S_{pf}$)**: This metric is the occurrence frequency of each extracted attribute-task pair. If an attribute-task pair appears very often across the text it implies that the article authors find it important. It should be noted that if a pair occurs twice within the same sentence, we only count it once. We normalize the frequencies between 0 and 1 with 1 indicating the most frequent pair and use this value as the frequency metric.
- **Sequence ($S_{ps}$)**: This metric is computed by observing the occurrence sequence of attribute-task pairs. If one task-attribute pair appears before another in a document it implies that the writer may find it necessary to evaluate the first pair before the second. If the user provides a corpus with multiple documents we first rank the pairs within each document based on their occurrence sequence. We then combine these rankings into an average ranked list. Finally, we normalize the ranks, with the highest ranked item receiving a value of 1 and the lowest 0, and utilize these values as the sequence metric.
- **Statistical Relevance ($S_{st}$)**: This metric is based on the statistical properties of the dataset itself and is independent of the text. By considering the statistical properties of a dataset we can reduce the importance of tasks that may appear across the text but might be irrelevant to the current dataset. Here we rank the attributes based on the statistical tests. Then for *correlation*, *clustering*, and *anomalies* we use the ranks generated by the respective statistical tests. For the remaining tasks, we use the maximum rank of an attribute across all statistical tests. We normalize the ranks between 0 (lowest ranked) and 1 (highest ranked) and utilize the values for the statistical relevance metric.

The final importance measure is computed as a weighted average of three metrics $I = w_1 S_{pf} + w_2 S_{ps} + w_3 S_{st}$ . The default values of these weights are set to (0.5, 0.2, 0.3). We chose to give a higher weightage to frequency as we believe that if an attribute or attribute pair and associated task occurs across documents frequently then it is referred to more frequently and thus is more important. Additionally, we discard any task-attribute pair that has a statistical score of 0. An importance measure of 1 would indicate that the task-attribute pair occurs at the start of a majority of the texts provided, it is also the most frequent pair to appear across all texts, and the task is also statistically supported by the data.

### *5.6. Mapping between Analytical Tasks and Visualizations*

TaskFinder communicates the ranking generated as a list of visualizations or charts that are appropriate for each attribute and task. We wish to support as many visualizations as possible to ensure that we can accommodate people with varying levels of visualization literacy. To achieve this, we studied various visualization guides produced by experts and corporations in the field of visualization and generated a list of visualization and task associations.

We collected and studied 19 visualization guides (see supplementary material for the list of guides). From each guide, we extracted the various tasks discussed and the visualizations suggested for each task along with the data constraints (data type and the number of items). We then counted the number of times a visualization was suggested for a particular task. The representation with the highest count would have the highest priority for the task while the representation with the lowest count would have the lowest priority. The results are shown in Table 1.

Our study revealed that their authors refer to six different high-level tasks or functions a chart can perform - *distribution*, *comparisons*, *part-to-whole comparisons*, *relationships*, *changes over time*, and *ranges*. We grouped *comparisons* and *part-to-whole comparisons* into a single parent task - *comparisons* - as the main difference between them is the type of data and not the function itself. Similarly, we view *changes over time* as a special type of *relationship* task where the relationship between an attribute's value and time is investigated. Additionally, the ten low-level tasks put forth by Amar et al. [4] can be mapped to these four high-level tasks. For example, some guides we studied state that the low-level tasks *correlation*, *clustering*, and *finding anomalies* tasks essentially require the user to investigate the relationship between data items and can thus be mapped to *relationships* task. Similarly, the low-level tasks *Determine Range* and *Find Extremum* can be mapped to the *range* task and the *Order* and *Filter* low-level tasks can be mapped to the *comparison* high-level task. The *Characterize Distribution* low-level task and *distribution* high-level task are identical. Finally, *Compute Derived Value* and *Retrieve Value* were not referred to in the guides. For these tasks, we chose to recommend the common charts - bar, line, and scatter plot.

### *5.7. Curating Visualizations*

Using the mapping of low-level tasks to high-level tasks and task-chart mapping we assign each ranked attribute-task pair a list of possible visualizations. We also set limitations on which charts can be assigned to an attribute based on its data type (e.g. line charts are only assigned to attributes with the time data type). At times, duplicates may arise due to attributes being paired with different tasks that require share a recommended visualization. These duplicates are merged and their importance measures are summed. Then the attribute-task-chart pairs are re-ranked based on the new scores and presented to the user.

The visualizations are presented as a list of cards via the interface shown in Figure 4. We prioritize bivariate representations over univariate representations as they are capable of providing the user with more information. Additionally, we give users the option to deselect any analytical tasks they are not interested in or any visualization they are unfamiliar with via a panel (left).

## **6. Demonstration**

We demonstrate TaskFinder across two distinct domains characterized by varying corpus sizes and quality. The two domains under consideration are automobiles and sports – highly discussed topics across the internet. This demonstrates its ability to identify attributes and their corresponding tasks and how they inform recommendations for visualization.

### *6.1. Car Comaprisons*

For our first demonstration, we use the Cars [30] dataset retrieved from Kaggle. The dataset has nine data attributes - *Model*, *Horsepower*, *Cylinders*, *Displacement*, *Acceleration*, *MPG*, *Weight*, *Year*, and *Origin*. As discussed, TaskFinder requires the user to provide text documents that discuss automobile analysis. Thus, we opted for car comparison reviews from online magazines. Our rationale stems from the belief that such reviews are inherently analytical in nature, as they systematically compare cars based on performance and features. Thus, we selected at random a set of 15 car comparison reviews for demonstration.

With the dataset and accompanying textual documents selected, we proceeded to upload them to TaskFinder, employing the interface in Figure 4. Our initial step involved selecting attribute types and refining the synonym list, as outlined in Figure 3. For instance, the *Model* attribute was set to the index type as it is unique and essentially a label for each data item and won't be considered during the recommendation phase. The attributes *Horsepower*, *Displacement*, *Acceleration*, *MPG*, *Weight* were set to numerical while *Cylinders*


**Figure 4.** The TaskFinder interface that takes in a dataset along with related textual documents via inputs on the top left and produces a set of recommended visualizations that can be used to explore the most important features of the data. Users can control the recommendations by selecting the tasks they are interested in and the visualizations they are familiar with via the task and visualizations panels on the left.

and *Origin* were set to ordinal and nominal. Notably, the *Year* attribute was set to the time type, although, in this particular case, TaskFinder treated it as an ordinal attribute due to the presence of multiple data points with the same year value. Next, we refined the attribute synonyms. As the words *Model*, *Weight*, and *Origin* are generic, they yield a larger set of synonyms with *Model* having the most - 17 synonyms. Given their generic nature, a substantial number of these synonyms were irrelevant within the automotive context and were consequently deselected. At this juncture, TaskFinder had the relevant input essential for generating visualization recommendations to explore the dataset.

TaskFinder analyzed the input and returned a list of visualizations (Figure 4). The textual articles contained sentences detailing various car specifications, coupled with references to analytical tasks. For example, consider the sentence "The MG has the higher power output – 170hp to the Tata's 140 (torque is an identical 350Nm at 1,750rpm) – but its wider, thinner-spread powerband means you have to shift less, and responses low down are actually a bit better." Here TaskFinder associated the term 'higher' with the *filter* task and the noun phrase 'power output' with the *horsepower* attribute. It also finds that there is a dependency (*amod*) between the two tokens thus inferring that the *filter* task was applied to the *horsepower* attribute. It also identifies '170hp' as an entity and associates it with the *horsepower* attribute and the *retrieve value* task.

As explained in Section 5.6, *filter* is a low-level task that maps to the high-level *comparison* task. The first part of the sentence refers to this task – a comparison of two cars in terms of their horsepower values. The second part is a *relationship* task, relating *powerband* to *shift frequency*. As these attributes are not present in the dataset, the sentence part will not be considered. TaskFinder systematically processes numerous sentences within the corpus, counting the number of attributes and associated task occurrences. It also ranks the attributes based on their order of appearance across the 15 articles. It then uses the occurrence count and appearance order along with the dataset statistics to produce the ranked list of visualizations shown in Figure 4.

Upon investigating the results we see that TaskFinder was able to recommend a list of visualizations. We see that it recommended some very frequently paired attributes. For example, it recommends investigating the relationships between *MPG* and *Horsepower*, *Horsepower* and *Cylinders*, *Horsepower* and *Acceleration*, *Horsepower* and *Weight*, and *Weight* and *Acceleration* as these relations were frequently reported in the articles. We compared these relationships to those investigated by the top 10 Kaggle notebooks (based on upvotes received) associated with the dataset and observed the same relationships being investigated by Kaggle users. However, we found that our system also recommended relationships that are not usually investigated such as the relationship between *year* and *Horsepower*, *Cylinders*, and *MPG*. The reason for this relationship being picked up was that at times the articles referred to cars by their model name and year ( e.g. "2019 Civic") and TaskFinder identified that these references to the *year* attribute were very frequent.

Overall, TaskFinder was able to use Car reviews to identify attributes and associated tasks to recommend appropriate visualizations. Most recommendations aligned with the attribute pairs and visualizations Kaggle users investigate in the same dataset. Thus with a set of relevant documents, TaskFinder was able to produce recommendations on par with a Kaggle analyst.

### *6.2. NBA Player Achievements*

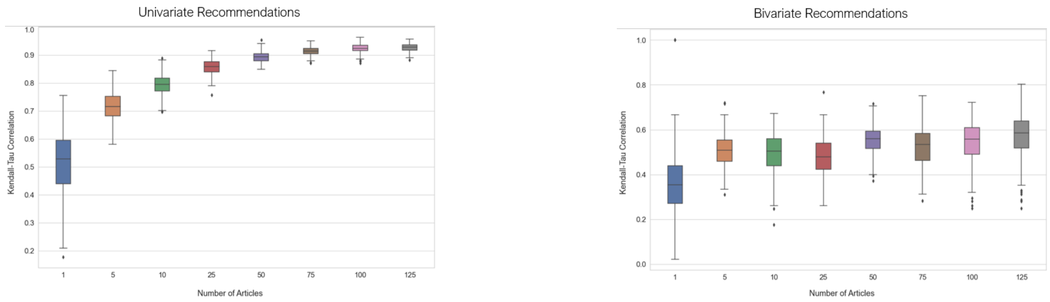
For our second demonstration, we focused on the sport domain. Specifically, apply TaskFinder to an NBA dataset [31] retrieved from Kaggle which had nine attributes of interest - minutes, points, rebounds, assists, steals, blocks, turnovers, fouls, age, height, position, and weight. We also provided TaskFinder with articles describing the achievements of historically great players sourced from the NBA website [41]. We believe that these articles were analytical in nature as they focused on players' career performances and compared them to other great players.

Just as we did with the cars dataset, we moved on to set the data types for the attributes and refined the recommended keyword set. The attributes were all numerical except for the player's name which we set to the index type. The attribute keywords TaskFinder found were all relevant thus we had to expend minimal effort in refining the keywords set. Having provided all the input necessary, TaskFinder analyzed the articles along with the dataset and provided a list of recommended visualizations.

Upon investigating the results, we the list contained visualizations that were primarily univariate. This is due to the fact that the articles are about "hall of fame" players and they tend to mention the statistically outstanding player performances in isolation. For example, sentences like "He also holds the all-time record for the highest field-goal percentage in a five-game playoff series", appear frequently in these articles. Here the *Find Extremum* task or *Range* task was associated with the points attribute referred to by the word "field-goal". The system found some bivariate relationships as well through sentences such as "In 80 games, he averaged 34.0 points and 11.4 assists." Here the *Retrieve Value* task was associated with the combination of the attributes points and assists, the number of games was ignored as it is not an attribute in the dataset.

Overall, associated the *Find Extremum* and *Retrieve Value* with the points, assists, and rebounds attributes. While the position attribute was only associated with the *Retrieve Value* task. Recommended bivariate attribute pairs included points and assists as well as rebounds and blocks. When compared to the top 10 Kaggle notebooks (based on upvotes received) associated with the dataset, the recommendations do not align well. A few Kaggle users investigated extremums on all the dataset attributes but most split the dataset into subsets based on position or NBA time periods (sets of seasons) to compare a subset of players.

While the results are not inline with what a Kaggle analyst may analyze, they are representative of what authors of the player profiles are interested in - a basic analysis focused on each player's best achievements. To get a more diverse set of task-attribute associations we would have to find a different source of textual information. In the sports domain today such kind of analyses are often reported in talk shows rather than articles. Thus, applying our method to a transcription of the talk shows would be an avenue for future investigation.




**(a)** **(b)**

**Figure 5.** We report the distribution of the Kendall-Tau correlation values between 25 recommendations generated over a set of $n$ articles for the cars dataset. We vary the value of $n$ and report each distribution above for each value of $n$ for both (a) univariate and (b) bivariate recommendations. From these distributions, we see that the correlations tend to stabilize when we have a total of 75 articles.

## 7. Evaluation - The Effect of Corpus Size

In this work, we hypothesize that if a corpus is large enough, we can extract the ranking that informs us of what attributes and tasks people are generally interested in for a particular data domain. Thus, we decided to study the effect of corpora size on the stability of a ranking generated for a particular dataset.

For this study, we use the Cars [30] and NBA player [31] datasets. We collected a total of 700 car comparison review articles and 20 NBA player profile articles. For each dataset, we select $n$ articles at random and compute the ranking. For each value of $n$ we repeat the process 25 times. We then compute the Kendall-Tau correlation between all rankings and report the mean distance and standard deviation. We repeat this process for different values of $n$. For the car dataset, we set the value of $n$ to 1, 5, 10, 25, 50, 75, 100, and 125. For the NBA dataset, we set the value of $n$ to 1, 3, 5, 7, and 10. We then investigate the distribution of correlation values to study at what corpora size the correlation stabilizes.

The results for the cars dataset are shown in Figure 5. Here we see that when we generate recommendations with just one article we have high variability along with a relatively lower median correlation value indicating that we can get very different recommendations if we base our analysis on just one article. As the number of articles increases, variability decreases and the median correlation increases especially when making univariate recommendations. We found that, for the cars dataset, the correlation between rankings stabilizes once we had at least 75 car comparison reviews.

## 8. Discussion and Limitations

**Impact of corpus quantity and quality.** Our method relies on information from a corpus to associate tasks with attributes and hence it is dependent upon the quality and correctness of the corpus' contents. Choosing a small number of texts, as in section 6.1, may not recommend the most frequent attributes and analytical tasks in the domain. For building general-purpose recommenders this is a limitation. On the other hand, curating a specific set of texts to form a corpus can be beneficial. For example, texts produced by a specific author or publication may extract a particular analysis style. Alternatively, an organization may choose to only use internal documents for recommendations thereby having some assurance of the corpus quality and analytical style.

We observed that our approach occasionally leads to trivial recommendations as it gauges what is of interest to the domain audience, which is not necessarily interesting for an analyst. For example, analyzing the NBA dataset led to a recommendation of simple univariate charts that compared players or teams. These top-ranked charts typically did not seek to explain certain relations, as bivariate charts often do. Thus in certain domains, textual documents might not contain all the interesting task-attribute associations. In such cases, we may consider methods to inform the user of the quality of the corpus or recommendations. For example, we can report a score based on the number of occurrences of attributes and associated tasks in the corpus. Alternatively, we could have wildcard recommendations [14] generated from underrepresented attributes or tasks.

Finally, we explored scientific domains like gene expression data, where attributes are rarely discussed outside academia. However, we faced two main issues. First, there weren't many accessible texts on these topics. Second, the ones we found focused on complex statistical analyses, not the kind of exploratory or basic analysis that TaskFinder was designed for. Additionally, the method based on standard synonym generators lacked the sophistication required to identify attribute references in text for scientific domains. Adapting TaskFinder to such data domains will require extending its capabilities to recognize complex tasks and specialized attribute names or references.

**Reliance on the user to provide relevant documents.** In its current form, TaskFinder requires the user to provide the corpus from which attributes and associated tasks are identified. While it removes the burden of picking the right attributes and analytics, it places the new burden of curating a corpus and ensuring its quality. This makes it less accessible. In the future, we would like to remove this burden from the user by automatically retrieving such documents. To fill this void one can explore crowd-sourced or web-crawler-based methods, to source analytical texts about various datasets or domains and create a knowledge base. Alternatively, we can explore methods to gauge the quality of user-provided corpora. In addition to the burden of finding articles, the user may need to refine keyword lists used to represent attributes, especially in cases where some attribute synonyms may not be applicable to the data domain. Finally, the recently emerging commodity large language models, embodied by ChatGPT [42] and the like, could form another source of textual information; prompts could be engineered in such a way that the returned text would reflect a certain viewpoint or target audience. One might even be able to capture some of the NLP analyses into the prompt.

**Beyond Attributes and Low-Level Analytic Tasks.** We focussed on the 10 low-level tasks as they are frequently used by other recommender systems making this work easy to pair with. However, our approach can be extended to other more complex tasks. Over the course of our investigation of articles, we observed that texts contain much more information that can be applied to the analysis of a dataset. One direction we explored was the evaluation section of research papers. These did not contain low-level tasks but they did mention attributes and statistical tests applied to them or chart types used to show them in figures. Such information is useful for recommending statistical tests and visualizations for scientific applications. Additionally, in our approach, we only recommend tasks for an attribute. However, texts mention specific attributes that refer to a subset of the data, for example, a specific car manufacturer or model, or a particular NBA season or player position. This information can be leveraged to identify subsets in the data that may be of interest to the analyst.

**Augmenting other systems.** We envision our approach of extracting attribute-task associations from texts to be a part of a recommender system rather than a standalone task recommender. With TaskFinder we paired our NLP extraction technique with a statistical analysis model and a visualization recommendation based on a collection of visualization guides. These components are interchangeable and it would be interesting to investigate if other systems such as Kopol [43] or Data2Vis [18] can be augmented with information gained form text and how they would perform. Our approach may also be of interest to researchers working on authoring tools that recommend appropriate visualizations to authors of articles. This is along the lines of the Kori system [44].

## 9. Conclusions

In this work, we developed a technique to recommend appropriate visualization tasks for a given dataset by extracting information from textual articles. To our knowledge, this is one of the first attempts at recommending visualization tasks specific to the domain of the dataset. We demonstrated via a case study that our technique could identify mentions of attributes and tasks in web-based texts and relate them to each other. Our approach builds on well-established methods from the fields of NLP and visualization, so we did not see an immediate need to perform a dedicated user study on our system; rather we show that the quality of recommendations is bounded by the articles provided. In the future, we intend to pair our work with newer NLP techniques and other recommendation strategies to build a more robust recommender.

**Author Contributions:** Conceptualization: Darius Coelho, Steve Greenspan, Serge Mankovski, Klaus Mueller, and Maria Velez-Rojas; methodology: Darius Coelho and Klaus Mueller; software, investigation: Darius Coelho; formal analysis, validation, Bhavya Ghai and Arjun Krishna; writing - original draft preparation, Darius Coelho; writing - review and editing: Darius Coelho and Klaus Mueller, funding acquisition, Steve Greenspan and Maria Velez-Rojas. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded in part by the NSF I/UCRC 1650499: Center for Visual and Decision Informatics (CVDI) Site at SUNY Stony Brook, CA Technologies, a Broadcom Company, USA, and NSF grant IIS 1527200.

**Institutional Review Board Statement:** Not applicable

**Informed Consent Statement:** Not applicable

**Data Availability Statement:** The text corpus employed for the studies as well as links to the visualization cheatsheets studied are included in the supplementary material.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| NL | Natural Language |
| NLI | Natural Language Interface |
| NLP | Natural Language Processing |
| POS | Part-of-Speech |

## References

1. Mackinlay, J. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics* **1986**, *5*, 110–141.
2. Roth, S.; Mattis, J.; Mesnar, X. Graphics and Natural Language As Components of Automatic Explanation. *SIGCHI Bulletin* **1988**, *20*, 76.
3. Casner, S. Task-analytic Approach to the Automated Design of Graphic Presentations. *ACM Trans on Graphics* **1991**, *10*, 111–151.
4. Amar, R.; Eagan, J.; Stasko, J. Low-Level Components of Analytic Activity in Information Visualization. In Proceedings of the Proc. IEEE Symposium on Information Visualization, 2005, pp. 111–117.
5. Saket, B.; Endert, A.; Demiralp, C. Task-Based Effectiveness of Basic Visualizations. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *25*, 2505–2512. https://doi.org/10.1109/TVCG.2018.2829750.
6. Bertin, J.; Berg, W.J.; Wainer, H. *Semiology of Graphics: Diagrams, Networks, Maps*; University of Wisconsin Press Madison, 1983.
7. Cleveland, W.; McGill, R. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal American Statistical Association* **1984**, *79*, 531–554.
8. Kerpedjiev, S.; Carenini, G.; Roth, S.F.; Moore, J.D. AutoBrief: A Multimedia Presentation System for Assisting Data Analysis. *Computer Standards & Interfaces* **1997**, *18*, 583–593.
9. Wills, G.; Wilkinson, L. AutoVis: Automatic Visualization. *Information Visualization* **2010**, *9*, 47–69.
10. Vartak, M.; Huang, S.; Siddiqui, T.; Madden, S.; Parameswaran, A. Towards Visualization Recommendation Systems. *SIGMOD Record* **2017**, *45*, 34–39.
11. Mackinlay, J.; Hanrahan, P.; Stolte, C. Show Me: Automatic Presentation for Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics* **2007**, *13*, 1137–1144.
12. Key, A.; Howe, B.; Perry, D.; Aragon, C. VizDeck: Self-organizing Dashboards for Visual Analytics. In Proceedings of the ACM SIGMOD, 2012, pp. 681–684.

13. Wongsuphasawat, K.; Moritz, D.; Anand, A.; Mackinlay, J.; Howe, B.; Heer, J. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* **2016**, *22*, 649–658.
14. Wongsuphasawat, K.; Qu, Z.; Moritz, D.; Chang, R.; Ouk, F.; Anand, A.; Mackinlay, J.; Howe, B.; Heer, J. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In Proceedings of the CHI, 2017, pp. 2648–2659.
15. Lee, D.; Setlur, V.; Tory, M.; Karahalios, K.; Parameswaran, A. Deconstructing Categorization in Visualization Recommendation: A Taxonomy and Comparative Study. *IEEE Transactions on Visualization and Computer Graphics* **2021**.
16. Luo, Y.; Qin, X.; Tang, N.; Li, G.; Wang, X. DeepEye: Creating Good Data Visualizations by Keyword Search. In Proceedings of the Proc. ACM SIGMOD, 2018, pp. 1733–1736.
17. Moritz, D.; Wang, C.; Nelson, G.L.; Lin, H.; Smith, A.M.; Howe, B.; Heer, J. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics* **2019**, *25*, 438–448.
18. Dibia, V.; Demiralp, Ç. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics & Applications* **2019**, *39*, 33–46.
19. Hu, K.; Bakker, M.; Li, S.; Kraska, T.; Hidalgo, C. Vizml: A machine learning approach to visualization recommendation. In Proceedings of the Proc. CHI, 2019, pp. 1–12.
20. Li, H.; Wang, Y.; Zhang, S.; Song, Y.; Qu, H. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Trans. on Visualization and Computer Graphics* **2021**, *28*, 195–205.
21. Wu, A.; Wang, Y.; Zhou, M.; He, X.; Zhang, H.; Qu, H.; Zhang, D. MultiVision: Designing Analytical Dashboards with Deep Learning Based Recommendation. *IEEE Transactions on Visualization and Computer Graphics* **2021**, *28*, 162–172.
22. Berger, M.; McDonough, K.; Seversky, L.M. cite2vec: Citation-Driven Document Exploration via Word Embeddings. *IEEE Trans. on Visualization and Computer Graphics* **2017**, *23*, 691–700.
23. Park, D.; Kim, S.; Lee, J.; Choo, J.; Diakopoulos, N.; Elmqvist, N. ConceptVector: Text Visual Analytics via Interactive Lexicon Building Using Word Embedding. *IEEE Transactions on Visualization and Computer Graphics* **2018**, *24*, 361–370.
24. Mahmood, S.; Mueller, K. Taxonomizer: Interactive Construction of Fully Labeled Hierarchical Groupings from Attributes of Multivariate Data. *IEEE Trans. on Visualization and Computer Graphics* **2019**, pp. 1–1. https://doi.org/10.1109/TVCG.2019.2895642.
25. Cox, K.; Grinter, R.; Hibino, S.; Jagadeesan, L.; Mantilla, D. A multi-modal natural language interface to an information visualization environment. *International Journal of Speech Technology* **2001**, *4*, 297–314.
26. Sun, Y.; Leigh, J.; Johnson, A.; Lee, S. Articulate: A Semi-Automated Model for Translating Natural Language Queries into Meaningful Visualizations. In Proceedings of the Proc. Smart Graphics, 2010, p. 184–195.
27. Gao, T.; Dontcheva, M.; Adar, E.; Liu, Z.; Karahalios, K.G. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In Proceedings of the Proc, UIST, 2015, p. 489–500.
28. Yu, B.; Silva, C.T. FlowSense: A Natural Language Interface for Visual Data Exploration within a Dataflow System. *IEEE Transactions on Visualization and Computer Graphics* **2020**, *26*, 1–11. https://doi.org/10.1109/TVCG.2019.2934668.
29. Narechania, A.; Srinivasan, A.; Stasko, J. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Transactions on Visualization and Computer Graphics* **2021**, *27*, 369–379. https://doi.org/10.1109/TVCG.2020.3030378.
30. Cars Dataset. http://archive.ics.uci.edu/ml/datasets/Auto+MPG. Accessed: 2022-11-29.
31. NBA Player Dataset. https://www.kaggle.com/datasets/drgilermo/nba-players-stats. Accessed: 2022-11-29.
32. Datamuse. https://www.datamuse.com/. Accessed: 2022-11-29.
33. Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. In Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Vol. 1, 2002, p. 63–70.
34. Clark, K.; Manning, C.D. Deep Reinforcement Learning for Mention-Ranking Coreference Models. In Proceedings of the Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2256–2262.
35. Rehurek, R.; Sojka, P. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* **2011**, *3*.
36. Speer, R.; Chin, J.; Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In Proceedings of the Proc. AAAI, 2017.
37. Wilson, B. The Natural Language Processing Dictionary. http://www.cse.unsw.edu.au/~billw/nlpdict.html. Accessed: 2023-08-29.
38. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In Proceedings of the Proc. NIPS, 2013, pp. 3111–3119.
39. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
40. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **2017**, *5*, 135–146.
41. NBA Legends Profiles. https://www.nba.com/history/legends. Accessed: 2022-11-29.
42. OpenAI. ChatGPT (Feb 13 version) [Large language model]. https://chat.openai.com, 2023.
43. Saket, B.; Moritz, D.; Lin, H.; Dibia, V.; Demiralp, C.; Heer, J. Beyond heuristics: Learning visualization design. *arXiv preprint arXiv:1807.06641* **2018**.

44. Latif, S.; Zhou, Z.; Kim, Y.; Beck, F.; Kim, N.W. Kori: Interactive Synthesis of Text and Charts in Data Documents. *IEEE transactions on visualization and computer graphics* **2022**, *28*, 184–194.