Interactive Exploration of Text Databases

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree
Master of Science in the Graduate School of The Ohio State
University

By

Ziqi Huang, B.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2015

Master’s Examination Committee:

Arnab Nandi, Advisor
Srinivasan Parthasarathy
Abstract

With the increasing importance of text analytics in all disciplines, e.g., science, business, and social media analytics, it has become important to extract actionable insights from text in a timely manner.

In this work, we take a close look at Interactive Data Exploration. We first introduce a novel interactive framework that allows social media analysts to tweak the text mining dashboards not just during its development stage, but also during the analytics process itself. Then we study the interactions between common users and the framework, and introduce a touch interface that help users to reach their search target in a efficient manner.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Vita</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 An interactive data analysis framework to support parameter sweeping</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Front-ends for interactive data searching and exploration by providing preference information</td>
<td>3</td>
</tr>
<tr>
<td>2. Interactive Tweaking of Text Pipelines</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Motivating Example: Twitter Analytics</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Challenges</td>
<td>10</td>
</tr>
<tr>
<td>2.3.1 Interactive Response Times</td>
<td>11</td>
</tr>
<tr>
<td>2.3.2 Text Operations as Blackboxes</td>
<td>11</td>
</tr>
<tr>
<td>2.4 The IntPipe Approach</td>
<td>12</td>
</tr>
<tr>
<td>2.4.1 Trading off Computation and Cache</td>
<td>12</td>
</tr>
<tr>
<td>2.4.2 Reusability</td>
<td>13</td>
</tr>
<tr>
<td>2.4.3 Opening Black Boxes</td>
<td>14</td>
</tr>
<tr>
<td>Initial Naive Bayes Classifier Model</td>
<td>15</td>
</tr>
<tr>
<td>Implementing Incremental Computation in Naive Bayes</td>
<td>15</td>
</tr>
<tr>
<td>2.5 Related Work</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Conclusion and Future Work</td>
<td>17</td>
</tr>
</tbody>
</table>
3. Beyond Binary Gestures: Exploring Documents using Touch Interfaces . 19

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.3 User Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
    Number of bins . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
    Placement of bins . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
    Bin selection gesture . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
3.4 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
3.5 Preliminary Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
    Participants . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
    Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
3.6 Conclusion and Future Work . . . . . . . . . . . . . . . . . . . . . . . . 25

Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Vita

January 17, 1988 ....................... Born - Zhejiang, China

2009 ............................... B.S.

2013-present ......................... Graduate Teaching Associate,
                               The Ohio State University.

Publications

Research Publications

Arnab Nandi, Ziqi Huang, Man Cao, Micha Elsner, Lilong Jiang, Srinivasan Parthasarathy, Ramiya Venkatachalam “Interactive Tweaking of Text Pipelines”. lecture notes in computer science, Volume 8999, 2015, pp 122-132


Fields of Study

Major Field: Computer Science and Engineering
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The IntPipe System for exploratory and interactive tweaking of text pipelines: User specifies an analytics dashboard as an explicit query, which is executed over a social media corpus and then presented as a visualization dashboard in the user interface. The user then explores and interacts with parameters triggering further queries, which are rewritten to leverage the intermediate result cache.</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Query tree representing the final example query.</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>BinGo Prototype Implementation</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>The satisfaction score. The first 10 users attended the paper exploration experiment and the rest 10 users attended the game exploration experiment.</td>
<td>24</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

As the rapid growth of internet, huge increasing information are posted online worldwide. Many traditional medias have shifted their focus to internet and built their own website to share news. Moreover, the well developed social network and crowd sourcing platforms, such as Facebook, Twitter and Yelp, trigger the popularity of sharing personal opinions. Those information are significant for discovering useful signals. For example, the feedbacks on amazon products provide information on both products quality and personal preferences. Combining with user’s profile, we can further analysis the popularity of products among different user groups.

Given this huge amount information published online, researchers are very excited about observing interesting information from it. Many studies have been published on machine learning algorithms to help analysts filtering useful information. Classification algorithm classifies objects to fixed number of classes and the clustering algorithm auto-groups different objects to different groups according to a certain measurement. However, as an end-user/analyst to these machine learning tools, there are several challenges for discovering useful information.
1.1 An interactive data analysis framework to support parameter sweeping

Many machine learning tools can be used to study the huge web-scale data. But before using the tools, a data analyst needs to know what to learn. For example, if you want babies to collect cubes from a mixer of cubes and spheres, you have to tell them what is the essential difference between these two objects — cubes have four corners and spheres have no corner. We call this essential difference a signal or a feature in the learning process. Therefore, choosing the right set of features can significantly improve the learning quality. In a proper learning process, analysts have to try different set of features and select the most effective one at the end.

However, only selecting signals/features is not enough for a good learning model. There are many ways to learn and the learning quality varies, choosing the best way is significant. For example, if you try to distinguish cubes and spheres, is tactile better than visual? The answer is different from one to another. Therefore, in order to improve the learning quality, analysts usually try many different features and algorithms to build the model by sweeping the parameters in the learning algorithm. There are many parameters in each algorithm and different parameter values can affect the performance of the learning algorithm. Proper parameters usually are provided by a parameter sweeping process in which researchers provide a list of parameters and its range to run and select the parameter values with the best performance on the test data. Therefore, it is really expensive to try out all the parameters and ranges.

Moreover, the performances for many parameter settings are not able to be auto-diagnosed. Therefore, human being must involve in the process and apply another parameter value according to the previous performance and personal analysis.
However, the execution time for a single iteration of learning is too expensive to support human and computer interaction. Many machine learning algorithms involve complex matrix computation that takes a long time. Some researchers might wait for several days to get the result for one parameter value combination and take several days to weeks to sweep different parameter values and select the best learning model. In this work, we provide a platform that helps users sweeping parameters efficiently and provides evaluation feedbacks in an interactive response time. Therefore, it allows data analysts to tweak the text mining dashboards not just during its development stage, but also during the analytics process itself.

1.2 Front-ends for interactive data searching and exploration by providing preference information

For a user without much knowledge on parameterized machine learning, it is really difficult for them to provide a proper subsequent parameter values after review the previous learning result. But, like teachers, human are really important in the process of choosing good features and parameter set for the machine learning algorithm. Therefore, we provide a platform that allows analysts to provide a simple feedback such as like/dislike to guide the learning process.

Many studies have been published for users to provide feedbacks. For example, scoring the search result is an easy and good measurement for the search quality. However, only evaluating the previous search result is not enough, we need more signals to guide the next learning process. Therefore, in this work, we are looking for a good way for users to provide feedbacks on previous learning quality, then improve it by regenerate the parameter values.
Moreover, as increasing amount of people tend to use portable devices for information searching. The feedbacks should be easy enough for users to provide feedbacks using a touch gesture. Therefore, we are looking for a way that can accept simple touch signals and provide information for our learning algorithm to sweep parameter setting. In such platform, users only need to provide a binary (like/dislike) feedback and our system will help them targeting to the best result efficiently.
Chapter 2: Interactive Tweaking of Text Pipelines

2.1 Introduction

Both the sciences and commercial enterprises are increasingly turning to analyzing textual data such as social media to enable them to act on intelligence gleaned from the data. This, in turn, has led to an increasingly complex set of analysis tools and frameworks [3, 17, 22, 41]. Given the scale of data in these contexts (e.g., for social media, over 500 million tweets are sent each day [46] and over 1.35 billion – almost half [20] of the adult Internet population – is on Facebook [18]), it is critical to rethink the text mining process in the context of exploratory analytics for efficiently surfacing time critical insights on text data.

An analytics task in this space is typically composed of a set of ordered and incremental text processing components to generate insights. A typical text analytics pipeline includes multiple stages of text preprocessing, model-building and evaluation, and aggregation. These pipelines can get fairly complex to manage as they often involve a stack of third party tools within a massive framework. Additionally, each step in the pipeline requires the tuning of several explicit and/or implicit (latent) parameters. The notion of “latent” is due to the fact that these parameters are tunable but not always exposed to the end user for exploration. For example, the
In this work we describe IntPipe, a framework that enables exploratory analytics through the interactive tweaking of text pipelines. The goals of our framework are orthogonal to existing analytics suites – we focus on the interactive aspects of tweaking text pipelines. Our framework exposes the different parameters (both latent and explicit) employed at each stage of the text pipeline to allow users explore and tweak.
2.2 Motivating Example: Twitter Analytics

Consider the problem that an analyst at a large movie theater chain might face: *what might the future sales of an upcoming movie might look like; and based on that analysis, how should rooms and showtimes be assigned?* She has access to tweets and wants to find what genres of movies are most talked about in order to derive insight into current trends. She first classifies the tweets to ensure they are related to the topic (in this case *movies*), extracts named entities from this subset of tweets using a Fuzzy Join\(^1\), followed by a join against the IMDB genre database to look up each movie’s genre. She then aggregates the remaining subset based on the genre of the movie. Finally, she builds a histogram representing the number of tweets of a particular genre in order to better visualize the result. This process can be declaratively represented by the SQL-style query:

**Query 2.1: Distribution of movie genres mentioned on Twitter**

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets) AS movie_tweets
JOIN entities
ON movie_tweets.id = entities.id
JOIN imdb ON entities.title = imdb.title
GROUP BY imdb.genre;
```

Going forward, the analyst may notice that there is an unusual distribution of tweets in a particular genre. She can then dive into this genre to manually inspect a list of tweets that have been classified under this genre and reclassify those that have been wrongly classified as movies. This reclassification will not change the overall query above, but will involve a change in the training data used for prediction and

\(^1\)We use the *fuzzy* or approximate string join [13, 24] as a standard operation, orthogonal to the contributions of this paper.
the resulting model is now used to reclassify the data. This *tweaked* version of the prior query can be represented as follows:

**Query 2.2: Retraining and reclassifying existing tweets**

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets, new_training_data) AS movie_tweets
JOIN entities
ON movie_tweets.id = entities.id
JOIN imdb ON entities.title = imdb.title
GROUP BY imdb.genre;
```

While performing the query above, the analyst may also notice that one reason for the unusual behavior is that in some cases tweets that have numbers (which are also names of movie titles) are wrongly included as movies. For example, a tweet whose text is “Let’s meet at 9” will be considered as a tweet containing the movie title “9” since “9” is a movie title. This can be fixed by changing a parameter in the title extraction of the stemming and preprocessing stage to exclude numbers.

This query can be represented as follows, the `is_number` controls whether we need to handle the movie whose name is a number:

**Query 2.3: Applying all previous queries and then handle movie titles as numbers**

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets, new_training_data) AS movie_tweets
JOIN entities
ON movie_tweets.id = entities.id
AND NOT entities.is_number()
JOIN imdb ON entities.title = imdb.title
GROUP BY imdb.genre;
```

By analyzing the sample tweets of a specific genre, she may discover that the reason for the incorrect movie title is that some movie titles were reformatted as hashtags. Hashtags, which users prefer to use to highlight words, are very popular in Twitter. Therefore, the analyst can unwrap the movie title in order to get the correct movie title. The query can further be modified as follows:
Query 2.4: Applying all previous functions and then unwrapping hashtags to movie titles

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets, new_training_data) AS movie_tweets
JOIN entities
ON movie_tweets.id = entities.id
AND NOT entities.is_number()
AND entities.hashtag().is_title()
JOIN imdb ON entities.title = imdb.title
GROUP BY imdb.genre;
```

By looking at the sample tweets for a specific genre, she may realize that some movie titles did not get extracted. This commonly happens in social media, due to the casual writing style. In order match the movie title better, the analyst needs to normalize both the tweets and movie titles in the IMDB (or any other movie database).

Query 2.5: Applying all previous functions and then change tweets to lower case

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets, new_training_data) AS movie_tweets
JOIN lowercase(entities) as entities
ON movie_tweets.id = entities.id
AND NOT entities.is_number()
AND entities.hashtag().is_title()
JOIN imdb ON entities.title = imdb.title
GROUP BY imdb.genre;
```

Now with these popular movie titles mentioned in tweets, the analyst may want to look into temporal trends, filtering movies by their release dates:

Query 2.6: Applying all previous functions and then filter movies by its release date

```
SELECT COUNT(movie_tweets.id), imdb.genre
FROM CLASSIFY(tweets, new_training_data) AS movie_tweets
JOIN lowercase(entities) as entities
ON movie_tweets.id = entities.id
AND NOT entities.is_number()
AND entities.hashtag().is_title()
JOIN imdb ON entities.title = imdb.title
AND year(imdb.date) = "2013"
GROUP BY imdb.genre;
```
Observing all of the queries listed above, we can see that each query is only a small modification of the original query, resulting in the final query tree shown below. Therefore, there are opportunities of reusing the results of the prior query. For example, Query 1.2 just needs to rebuild the classifier model and relabel all tweets, then filter the tweets that have been labeled as 1 – there is no need to re-do the normalization, title, and feature extraction steps.

Given the motivation and running example, we now envision an interactive dashboard for text analytics. The following section describes the challenges faced when building such a framework, and Section 2.4 details IntPipe, our system.

2.3 Challenges

Motivated by this example, the vision of IntPipe is to provides an interactive dashboard for text analytics which can respond to the end-user within interactive times. To this end, we list two key issues faced with this vision.
2.3.1 Interactive Response Times

Empirical evaluations in human-computer interaction have demonstrated that increased response times influence data exploration behavior [27], and that response times in the 100-1000ms window [31,42] allow interfaces to seem perceptibly “instantaneous” with respect to end-user interaction. Pipelines, and the following analytical steps are strictly batch operations, typically performed in a one-time setting. After the initial pipeline creation phase, there is little support to iteration and interaction with the text pipeline, or tweaking various parts of the text processing, performing subtle but significant changes. For a given query, each combination of parameters represents a unique processing of data. Given that the interface exposed to users is expected to be *interactive*, each change in query or tuning parameter should affect a change in the analytics output in low latency. In the simple case, however, a change in the query, or any one of the parameters triggers a re-computation of the entire analytics pipeline. Given the possible resource requirements, complexity of each of the steps involved, and the parameter space, analyzing data at scale by recomputing the entire pipeline is thus untenable.

2.3.2 Text Operations as Blackboxes

Tools for natural language processing that form the core of the pipeline are typically black boxes. The underlying mining process is not transparent to end users. With large scale, ad-hoc text analytics coming into play, this text mining process can become slow and arduous. Thus, exploration of various parameter settings will be time-consuming as the entire pipeline has to be re-executed every time for every follow-up query involving a new parameter setting. Also, the opaque nature of
such functions makes it impossible to perform query optimizations such as predicate pushdowns. It is thus necessary to expose the parameters of these black boxes and make them configurable by providing a dynamic dashboard for interactive exploration.

As a solution to these challenges, we propose to expedite this process, and allow analysts to interactively modify large-scale text pipelines. Enabling such interactivity has several compelling uses. End users can tweak and overhaul text pipelines not just during the inception stage of the pipeline, but during the analytics process itself. Further, it allows for interactive ad-hoc analysis of text and the ability to modify the pipeline to correctly articulate the overall analytical intent. Finally, it allows for quick “what-if” analyses of the data, testing several hypothetical assumptions on the dataset.

2.4 The IntPipe Approach

Based on the motivating example and the challenges at hand, we propose three methods that enable the interactive tweaking of text pipelines, collectively named IntPipe. As shown in Figure 2.1, IntPipe models queries as DAGs (directed acyclic graphs) of text processing functions, and leverages an intermediate result cache to speed up text analytics queries.

2.4.1 Trading off Computation and Cache

A key observation of the analytics process is that follow-up queries are typically a slight modification of the prior query. Thus, based on this interaction pattern, there will likely be several opportunities for result reuse. Reusing intermediate results allows us to drastically reduce the execution needs of each analytics pipeline, thereby
making interactive response times possible. There is however, an important caveat: caching all the intermediate results can be prohibitively expensive (e.g., one version of the entire corpus per function node or more, if the function inflates the data, such as shingling).

We are thus left with an interesting problem: given a particular query pipeline, for which of the subqueries should we cache intermediate results, and which should we discard? Our insight into this problem is to consider all the possible transformations of the current query, and cache subqueries which provide the most savings of execution time for this transformed query, given the least additional space requirement. We model this problem as an optimization problem for a query $D$ as follows:

$$\arg\max_{\text{subqueries} \in \text{POWER-SET}(D.\text{subqueries})} \sum_{D' \in \text{TRANSFORMS}(D)} \text{Benefit}(D')$$

where:

$$\text{Benefit}(D') = t \cdot \text{Speedup}(D', \text{subqueries}) - d \cdot \text{Size}(D', \text{subqueries})$$

Here $D.\text{subqueries}$ represents the set of subqueries in the query, and $\text{TRANSFORMS}(D)$ represents the set of queries that can be constructed by modifying the current query $D$, either by changing a parameter or the structure of the query. $\text{Speedup}$ and $\text{Size}$ represent the time benefit estimate and the data usage estimate for caching the additional intermediate results at a subquery. Estimators and parameters can be empirically derived by profiling the functions over samples of the data.

### 2.4.2 Reusability

Unlike traditional result reuse, text pipelines have an interesting property that needs elucidation. The output of certain text transformation functions can be reusable
with respect to another function, i.e., we can materialize the result of queries with old parameter values and reuse it to evaluate the query with new parameter values. For example, consider \texttt{DISTINCT(TO\_UPPER(entities))}. Should the analyst decide that the \texttt{TO\_UPPER} be changed to \texttt{TO\_LOWER} for some reason, instead of triggering a full re-execution, it is possible to \textit{reuse} the prior result, and simply return \texttt{TO\_LOWER(DISTINCT(TO\_UPPER(entities)))}, which is logically equivalent since \texttt{TO\_LOWER} and \texttt{TO\_UPPER} are reusable for each other. This optimization is particularly useful for large collections with a small number of distinct elements. Upon recognizing such an opportunity, IntPipe can trigger a query rewrite based on the result cache contents, drastically reducing response time. It should be noted that not all functions are reusable – thus the pairwise compatibility of functions will need to be considered at the query rewrite layer.

### 2.4.3 Opening Black Boxes

A key challenge mentioned in Section 2.3 is that some text operations are typically viewed as \textit{black boxes}, not amenable to optimizations such as cache reuse. To show that we can indeed make text operations amenable to optimizations, we demonstrate simple modifications to \textit{one} such black box – a Naive Bayes classifier – that allow for caching and incremental use. We pick Naive Bayes as an example since it works well in several domains, its performance and robustness are well understood \cite{9,30,32,38}, and since incremental variants have been widely studied \cite{11,26,34}. Other classifiers and faster methods are heavily motivated \cite{8,10,12,19,32} and can be similarly modified to reuse the intermediate result cache.
The probabilistic model for a Naive Bayes classifier is a conditional probabilistic model:

\[ P(C|F_1, \ldots, F_n) = \frac{P(C) \prod_i P(F_i|C)}{P(F_1, \ldots, F_n)} \]  

(2.3)

More formally, given \( P(C) \), the probability of two classes: movie tweet / non-movie tweet, and \( P(F_i|C) \), the conditional probability of one feature value given the class, we can compute the probability of \( P(C|F_1, \ldots, F_n) \), the conditional probability of one class given the feature vector.

**Initial Naive Bayes Classifier Model:** We now consider caching opportunities within the classifier, when implementing the classifier inside a relational database. The basis of a Naive Bayes model involves maintaining a histogram of feature occurrences in the training set, a relatively fast database aggregation operation. The speed also allows for easy updates, in case the user supplies additional new training data. Further, a database implementation allows for the persistence and reusability of intermediate and final tables created for classification.

**Implementing Incremental Computation in Naive Bayes:** Since a Naive Bayes Classifier is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions, for each new training data with feature vector \( F = (F_1 = 1, F_2 = 0, \ldots, F_n = 0) \) and \( C = C_1 \), we merely need to update the count of \( P(F_1 = 1, C = C_1) \), \( P(F_2 = 0, C = C_1) \) and all feature values in the feature vector. However, a small update of the count will not change the final result. Therefore, we set a threshold to filter out the small updates of count and only change the count in the likelihood table when the update is larger than the threshold. Since our use case is that of social media where tweets have a 140-character limitation,
feature vector of each tweet will be very sparse. We expect the number of changes in
the likelihood table to be small, and thus the number of database (and cache) updates
will be small. This can be implemented by setting up a chain of database triggers that
batch count updates from the likelihood table to a hash value table to the prediction
table. Changes are propagated once the updates cross pre-fixed thresholds.

2.5 Related Work

Visualization-based analytics dashboards for text and social media have gathered
increased attention [2,3] lately. The IntPipe framework explores making such systems
more interactive and tunable, by introducing parameterized exploration of the under-
lying text mining stack itself. Frameworks on social media data have investigated a
variety of text-oriented functions beyond typical database processing, such as location
analysis [41], semantic analysis [22], topic analysis [37], and prediction [4]. Olston et
al. [35], have looked into the iterative analysis of web-scale data using query templates,
utilizing a combination of offline and online computation given a query workload. Our
framework utilizes a similar formalism and enables the interactive exploration of such
analytics by speeding up execution. Marcus et al. [29] introduce a stream-oriented
query processing system for Twitter events. Ideas from our work can significantly
improve the iterative refinement user flow of such a system.

The concept of recycling of intermediate results has also been employed in several
different aspects of databases unrelated to text analytics, such as column stores [21].
From the caching perspective, there exists a significant body of work in query rewrit-
ing using materialized views [15,28,36]. Gupta and Mumik [14] provide the theoretical
bounds for selecting views to materialize under a maintenance cost constraint, which
we can use as a basis for our optimization: [39] investigates a similar determination of additional views to be materialized as an optimization problem over the space of possible view sets. Roy et al. [40] demonstrates the practicality of heuristic-based, multi-query optimization. These studies above are based on standard SQL functions; our framework additionally considers user-defined functions, and leverages reuse properties specific to text mining pipelines.

2.6 Conclusion and Future Work

In this paper, we introduced IntPipe – a framework motivated by the growing use of text analytics dashboards. IntPipe allows parameterized tweaking of text processing pipelines, by modeling them as DAGs of possibly reusable functions. By leveraging intermediate result reuse and query rewriting, IntPipe enables fast and iterative processing of text pipelines.

Going forward, the vision for IntPipe can be extended in three fronts. First, it would be useful to handle not just changing text pipelines, but also continually changing data, in the form of streaming queries. The provision of result reuse in this regard is challenging, given that intermediate results will need to be invalidated as the stream of data passes. This can be made possible by efficient use of database triggers. Second, we observe that the analyst is often faced with too many parameters and knobs to tweak. In this light, in addition to providing interactive access to tweaking pipelines, the system could also guide the user to possibly ideal parameter settings. This can be done by speculating, precomputing, and caching parameter combinations ahead of time. Third, following the pattern demonstrated with the Naive Bayes classifier, the framework could extend to support more complex text processing functions such
as named entity recognition, clustering and advanced classification methods, most of which are still considered blackboxes from a data processing infrastructure perspective.
Chapter 3: Beyond Binary Gestures: Exploring Documents using Touch Interfaces

3.1 Introduction

Nowadays, mobile devices become more and more popular for reading, gaming and even business because of their portability and computational power. However, searching and exploring content on mobile devices is quite different from PCs. Traditional document exploration techniques provide recommendations according to the keywords that users provide. When the search engine shows a list of results, users will check the result list to either modify the keyword and start another search or stop searching. This approach makes the exploration more difficult on touch devices since users need to regenerate keywords multiple times to get the target results and typing is expensive on touch interface. Therefore, avoiding multiple inputs of keywords is essential for mobile devices users.

Gesture-driven interfaces have become popular with portable, keyboard-less computational devices. The success of mobile application such as Tinder [1] – a dating application in which users can swipe the recommendation photo left to dislike it and right to like it – highlight the user demand for gesture-driven interfaces. However, providing recommendations for text is different from photos since users are able to
express a preference on pictures in a glance, but expressing a preference on documents takes time for reviewing. Therefore, it is important to minimize the number of gestures required from the user.

In addition, the lack of keyboard increases the difficulty of users to express their query. Hence, providing interesting words to guide the user to the target document is extremely valuable. Moreover, it is important to not only lead users to the target document, but also facilitate exploration of new interesting documents.

To adapt gesture-driven interfaces to text recommendation and exploration, we introduce BinGo. On each recommendation, in addition to the gesture for swiping to the left or to the right to indicate ‘like’ or ‘dislike’, BinGo allows users to drag the document to one of the bins that we provide to indicate the reason for liking or disliking a document. In BinGo, the reason bins are shown as a set of representative words in the document and aim to give users a better preview of the document. In the meantime, our reason bins can also guide users to their target document. Moreover, exploration of document is supported in BinGo by providing the words that have a broader connection with other documents.

3.2 Related Work

Allowing users to effectively specify keyword and textual queries is an important challenge. Adaptive search suggestions for library portals [25] and recommendation for related queries for Web search have been developed over the past decade [5].

As users move to mobile devices for a majority of their searching and information seeking tasks, keyword input is cumbersome and time consuming. Based on the increasing amount of mobile devices, which have a large touch screen and no keyboard,
sold in the world, the use of gesture in such interface is clearly a popular interaction method. Searches using mobile devices are performed in rushed and ad-hoc settings, which often exacerbates the “tip-of-the-tongue” problem in query specification. Thus, in an interface where the user cannot express their query, or is not familiar with the data or schema of the dataset [7], prompting the user with suggestions is extremely necessary. Faceted search [45] and browsing [16] has been a popular solution [23] to this problem. It is also possible to minimize the number of decisions by modeling the query session as a multi-way search tree [6], or using a cost-based approach [43]. In contrast to the hard search problem, BinGo blurs the line between query recommendations [44] and result exploration [33], while providing a convenient and usable interaction method for gestural interfaces.

3.3 User Interface

BinGo presents the top recommendation document in the center and the next top seven as clickable tabs at the top of the screen. Clicking on the tabs on the top, the corresponding full document will be shown in the central pane. After reviewing it, users can swipe the current document to left if they dislike the document or right if they like it. BinGo shows the previous reviewed document on either side of the dislike/like bars to indicate previous decisions. Moreover, reason bins are triggered by tapping, scrolling or swiping gestures. The user can swipe the document through one of these bins that are located on either side along the central document.

Number of bins  Motivated by studies on human cognitive capacity [47], only the top-5 bins are shown for users to select. Bins are shown when user trying to make a preference, users can chose any of the bins at any moment when they reviewing
the document. With a long list of bins, users need to recheck them whenever they forget one of them, hence increase the total query cost. Moreover, the relatively small screen size of mobile devices does not allow more than five to seven bins.

**Placement of bins** The location of reason bins is decided by the usage of bins in the system. In BinGo, bins are used to indicate the reason to like or dislike the current document, placing them as the gates to the dislike/like bars makes users understand the bins easier.

**Bin selection gesture** In the gesture-driven interface, binary gestures such as “Like/Dislike” can be implemented using a swipe gesture. However, when users are required to select a reason bin, more gestures are required since they need to drag the document to a specific location to indicate specific bins. To avoid extra gesture costs in our bin selection interface, BinGo detects the selected bin by checking whether the

---

**Figure 3.1: BinGo Prototype Implementation**

- Users use a history of documents they disliked/liked and query terms built from swiping through reason bins.
- Catchall bins allow users to like/dislike a document without reason.
- Placement of bins: The location of reason bins is decided by the usage of bins in the system. In BinGo, bins are used to indicate the reason to like or dislike the current document, placing them as the gates to the dislike/like bars makes users understand the bins easier.

- Bin selection gesture: In the gesture-driven interface, binary gestures such as “Like/Dislike” can be implemented using a swipe gesture. However, when users are required to select a reason bin, more gestures are required since they need to drag the document to a specific location to indicate specific bins. To avoid extra gesture costs in our bin selection interface, BinGo detects the selected bin by checking whether the
bin is on the path of swipe gesture. Therefore, under the same gesture constraint, our system allows users to not only present their preference but also indicate the reason.

3.4 Implementation

BinGo is implemented as a web application with its frontend written in JavaScript and HTML, and its backend server written in Python.

The documents are modeled as bag of words. We support the vector space document retrieval model where the bin candidates are terms with a high term-frequency-inverse document frequency (tf-idf) score. It helps to distinguish the current document from others since it provides terms that have high frequency in the current document but low frequency in the collection.

Then we eliminate the terms that have been previously selected. Using the vector space model with cosine similarity, BinGo determines the relevancy of other documents. To support document exploration, we also consider the words that have broader connection to other documents. Since the purpose is to refine the search query, BinGo only consider the terms that appear in a certain number or more documents as candidates.

3.5 Preliminary Results

We perform a preliminary user study to analyze the usability and impact of recommendation quality of BinGo. We use a within-subjects study design and look to rule out catching-on effects and other biases.

Participants We recruited 20 participants: 10 subjects were graduate students who are familiar with the paper review process since one of the application in user study
Figure 3.2: The satisfaction score. The first 10 users attended the paper exploration experiment and the rest 10 users attended the game exploration experiment.

requires users to review academic papers; the remaining 10 subjects were students who are familiar with video games. None of the subjects had used BinGo before.

Dataset Two datasets are used in the user study: the VLDB dataset includes 2035 papers from VLDB that published between 1975 and 2013 and the GiantBomb games dataset includes 44150 video game descriptions collected from GiantBomb.com.

Method For each user study, we first show the users a demo of searching papers / games using the two interfaces: SWP – swiping interface without reason bins and BinGo – swiping interface with reason bins. Then the users were asked to evaluate the usability and recommendation quality by providing scores from 1 to 10. 1 represents weak satisfaction while 10 represents strong satisfaction.
Figure 3.2a shows the user satisfaction on the recommendation quality of both interfaces and datasets. A t-test is applied and the small p-value(<0.0001) shows the recommendation quality of BinGo is significantly better than SWP interface.

Figure 3.2b shows the user satisfaction on usability of both interfaces and datasets. We notice that BinGo achieves weaker satisfaction than SWP interface. However, a t-test is applied and the p-value(0.9702) shows no significant difference on usability between two interfaces. Thus, BinGo provides a better exploration experience with insignificant changes to the user experience.

3.6 Conclusion and Future Work

In this paper, we recognize that text-related operations are hard to perform in mobile devices. Beyond simply binary gestures for “like/dislike” actions that allow users to rapidly sift through documents, reason bins allow for better document previews, and also following their changes of interest. By integrating reason bins into a swipe-based gesture, users are able to articulate the reason for liking or disliking a paper in a fluid and efficient gesture.

Based on the current implementation, there are several interesting avenues of future work. From a text database querying standpoint, alternative signals can be used for bin generation. For example, the time for users to read different section/pages of the document can be used to weight the term in a specific section. Considering user interest model is another reasonable extension to better guide users to their target document.

From a user interface standpoint, there are several chances to make the interface more usable, by considering different visual layouts for the bins depending the weight
of each bin. In our current interface, all bins are aligned and in the same size. However, because they have different weight, fisheye menus could further indicate the weight ratio among all bins.
Bibliography


