TagRefinery: A Visual Tool for Tag Wrangling

Christoph Kralj
University of Vienna
Vienna, Austria
christoph.kralj@gmail.com

Mohsen Kamalzadeh
Simon Fraser University
Burnaby, Canada
mkamalza@sfu.ca

Torsten Möller
University of Vienna
Vienna, Austria
torsten.moeller@univie.ac.at

ABSTRACT

We present TagRefinery, an interactive visual application aiding the cleaning and processing of open tag spaces, such as those in Last.fm or YouTube. Our pre-design analysis showed a need to support a spectrum of user expertise from novice to advanced, which resulted in two distinct interface modes. Summative evaluations of TagRefinery showed that it could effectively guide the novice users through the workflow by giving them brief but helpful explanations on why each step was required, and providing visual and statistical aids to help them in making important decisions. This is while our more expert users greatly appreciated the amount of control and granularity over the workflow that our more advanced interface mode offered. Both the underlying tag cleaning workflow and the interface were designed iteratively in a participatory design process in collaboration with research on a music recommendation interface based on Last.fm tags.

ACM Classification Keywords

H.3.3. Information Search and Retrieval: Selection process; H.5.2. User Interfaces: Graphical user interfaces (GUI), User-centered design; I.2.7. Natural Language Processing: Text analysis

Author Keywords

Data Wrangling; data cleaning; social tags; folksonomy; visual data analysis; user centred design; graphical interface

MOTIVATION

With the rapid rise of online content consumption and subscription services, the range of choices for the user has become massive and rather intimidating. This is true for almost any digital content, such as movies, music, academic papers, and books. One key ingredient of many such libraries is the tags or keywords made by users or experts. These tags can help us find both information that we are looking for, and new interesting content that we may not actively seek. Tag spaces, which are often called folksonomies, can be diverse and rich on information, and have been utilized in various areas such as information retrieval interfaces, recommendation algorithms, and music listening history analysis [2, 12, 13, 27, 28]. However, they can also suffer from various issues that are caused by their open and free-form nature. The greatest of these issues is noise, which includes spelling errors, subjective terms (such as "awesome" and "favourite"), and other tags which do not describe the content in a useful way (such as "seen live" in case of music). While a body of literature has previously concentrated on understanding the semantic structure of social tag spaces [16, 17], there is a lack of procedures and interfaces targeted at aiding researchers in cleaning such datasets. In this paper we present an interactive graphical user interface plus a novel tag processing workflow that aim to simplify the process of cleaning open tag spaces, while retaining as much information from the data as possible.

Depending on the intended use of the cleaned dataset, different researchers might have varying standards for the overall quality of the output. For instance, if tags are being cleaned for use in recommendation algorithms and interfaces, one would need a dataset devoid of subjective tags as they lack useful descriptive information. Meanwhile, such tags might be essential in behavioural research on why people identify items as their favourites. As such, a full automation of the process, while desirable, appears out of reach. In TagRefinery, the user plays a central role in cleaning the tag space by providing key inputs at various stages.

TagRefinery's workflow was born out of our need to clean the Last.fm folksonomy for our colleagues who worked on a music recommendation project [12]. As we continued revising our workflow, we saw the necessity of being able to quickly visualize aspects of the dataset in order to aid the process of choosing various important parameters in the workflow. This led to a participatory design practice, in which we solicited from them frequent feedback on both the underlying workflow and the graphical interface. In all stages, external users were also recruited for more impartial formative and summative evaluations.

As researchers with an interest in utilizing tags do not necessarily posses knowledge of text cleaning and natural language processing techniques, the application caters to a broad range of expertise by providing two different interface modes: *Guided*, and *Advanced*, plus one overview and experimentation screen called the *Linked* view. The Guided mode, which targets novice users, walks the user through all steps of the workflow and provides short explanations of why each step is performed. Our Advanced mode presents a structural view, facilitating quick access to all parameters of the workflow

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI'17, May 06 - 11, 2017, Denver, CO, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025868

and more detailed configurations. Finally, the Linked view presents an overview of the parameters of the most crucial steps of the workflow, enabling quick experimentation on these values and how their changes affect the output.

Summative evaluations were carried out from two points of view: (a) usability, and (b) usefulness. Our results indicated a high degree of usability (an average SUS [5] score of 80.83). All our participants felt the interface was intuitive and easy to follow. For the second study, we performed two case studies with participants whose work involved processing of various kinds of text data, which while similar to folksonomies, had key differences that could help us understand how well TagRefinery can adapt to use cases other than cleaning social tag spaces. These participants helped us identify a number of key functionalities for future improvement. In addition, all participants mentioned that they did not know of a tool that supported the same tasks and workflow as TagRefinery.

In summary, the contributions of this study are:

- We report on the iterative design process of a graphical interface for cleaning and filtering folksonomies, discuss the lessons we learned, and provide design considerations for future research on tag cleaning interfaces.
- We identify the distinct needs of our potential users based on their level of expertise in text processing techniques. We also discuss the different usage behaviours of novice vs. experienced users, and examine whether each interface mode of TagRefinery proved appropriate for its intended users.
- We introduce a novel workflow for cleaning tag spaces, upon which the above interface is built, and report on the outcomes of cleaning the Last.fm folksonomy in collaboration with research on a music recommendation interface.

RELATED WORK

Our contributions relate to several fields of research. In this section we discuss how our approaches differ from or build upon previous work in each of these fields.

Information Retrieval

Folksonomies are powerful sources of data in information retrieval. Contrary to fixed taxonomies, which might lack sufficient categories and terms to accommodate all items [10], social tag spaces lack a hierarchical structure, and thus give the users the freedom to fully express their needs and views [18]. Previous studies have confirmed the descriptive power of social tags in various domains with encouraging results [24]. In the music domain, Levy and Sandler [17] analysed user tags from Last.fm, and found that despite their noisy and subjective nature, the tags defined a low-dimensional semantic space that is highly effective in capturing music similarity. Despite these, as mentioned earlier, folksonomies suffer from several issues. Lamere [15] discusses some of these issues with regard to the Last.fm folksonomy. One of the most prominent problems is the immense amount of noise that open social tag spaces suffer from. Our approach focuses on cleaning noise from tag spaces, while saving as much information as possible on the annotated items. While the above mentioned studies and many more have focused on the semantic structure and descriptive power of tag spaces, few have contributed to ways of cleaning and filtering them. In the mentioned studies, tag datasets are typically only processed through a simple occurrence-based filter that removes tags which appear rarely; a method that leads to much loss of information in the removed tags.

Interfaces and Visualization

To the best of our knowledge there are no previous tools or techniques that target the tasks supported by TagRefinery. The most similar tools are Wrangler [14] and OpenRefine [1]. Wrangler facilitates easy and computer suggested data transformations (e.g. regular expressions) and utilizes semantic data types for validation and type conversion. From a workflow perspective, Wrangler is used right before TagRefinery to fix issues such as missing values and malformed cells, and to transform the data to the proper format for TagRefinery. However, it does not provide a step-by-step workflow for cleaning folksonomies. OpenRefine is an open source application for cleaning data which provides the classic spreadsheet view. While it has some features in common with our algorithms, such as clustering based on word similarity, it is not catered to any specific workflow and requires a high level of expertise from the user. Some more complex operations even need user created scripts necessitating an affinity for coding, which we did not want to require from users. In comparison, TagRefinery is carefully crafted for the specific purpose of folksonomy cleaning.

Natural Language Processing

Correcting spelling errors is one of the most critical parts of our workflow. Ringlstetter et al. [20] showed that web documents suffer from an impressive amount of spelling errors. Most studies in the spell checking literature apply the noisy channel model to fix errors. This model consists of two parts: a source (prior) and a channel (error) model to account for spelling transformations on letter sequences [4]. Different models are mostly distinguished by the error model. Examples include the use of pronunciation similarities [25] or generic string to string edits [4] to describe the error model. Another approach is to search for correct alternatives by semantic distance based on WordNet [11]. The noisy channel spell checkers and the WordNet method are not suitable for our purposes as they are based on dictionaries to be used as ground-truth. That would however limit our ability to detect meaningful but niche and completely new words. In addition, the fact that tags generally lack a grammatical structure [10] significantly reduces the effectiveness of more sophisticated approaches such as those based on Hidden Markov Models, like the one by Fossati and Di Eugenio [8]. Our approach is statistical and uses the available information on the popularity of tags in the dataset along with user input to decide which tags are likely to be correct. This makes our technique independent of language, which is crucial for open folksonomies.

[1] http://openrefine.org/

Tags often contain multiple words. These are called Multiword Expressions (MWEs) in the natural language processing community and detecting them plays a critical role in our workflow. MWEs are an important part of our languages and are difficult to tackle [21] in all NLP problems. Associative measures like co-occurrence patterns are a common way of identifying MWEs [6, 7]. There are a number of works that combine associative measures and semantic properties [7, 26] or statistical measures and alignment-based information [9, 19] with prediction models to identify MWEs. Supervised models have also been proposed recently for identifying MWEs [22], however, such approaches do not fit our case as we have no training data. We use co-occurrence and statistical measures [6], which are not dependent on language.

TAGREFINERY – DATA AND WORKFLOW

In this section we first discuss the Last.fm data, which was our main test dataset through-out the design of TagRefinery. We then describe our workflow and explain why each of its steps can be crucial to achieving a clean output and retaining as much information as possible from the noisy tags. In the next section, we dive into our discussion on TagRefinery's interaction design by reporting on our pre-design analysis, design requirements, and design choices.

Data

The Last.fm dataset, which we formed through mining tags for one million songs[2], was the main dataset based on which we developed our techniques[3]. Many social tag datasets come with meta-information on the tags and annotated items. For instance, the Last.fm API provides the percentage of taggers who applied a tag to a song, the number of listeners for each song, and the number of times a song has been played. If such meta-information exists, it can be used to generate a preliminary *quality* measure for each of the tags, which is then utilized in various stages of the pipeline. Computation of this *quality* measure should be done manually before importing the data into TagRefinery. As an example, the user could precompute TF-IDF values and use those as *quality* measure. If no such measure is imported, the tool uses the number of items each tag appears in as a replacement. The formula for the specific *quality* measure we used for the Last.fm dataset is described in Appendix A. To avoid confusion, we will always use italic type for referring to this measure.

Workflow

Figure 1 shows a diagram of the underlying tag cleaning workflow. There are three high-level phases: (a) Pre-Filter, (b) Spelling and Multiwords, and (c) Polish and Salvage. These span nine more granular steps that are discussed in detail in Appendix B. We settle with a brief explanation of the workflow here.

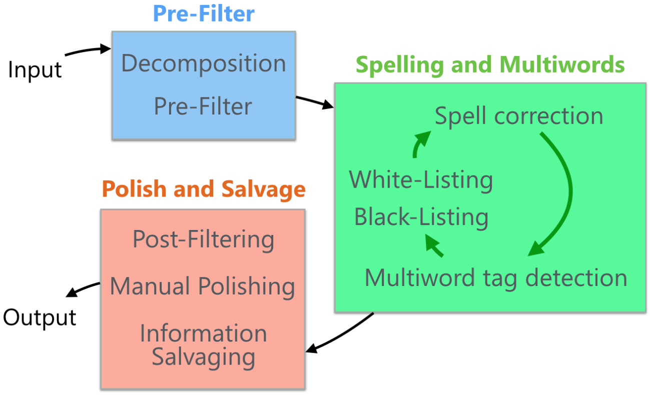


Figure 1. TagRefinery's workflow for cleaning social tags.

1. **Pre-Filter:** Since tags can contain numerous words, all tags are first decomposed into single words in order to perform spell correction later[4]. With larger datasets (hundreds of thousands of tags), the complete workflow can take hours to run. Hence, the Pre-Filter step can be used to temporarily limit the set of words to the ones with the highest number of occurrences, which will act as representatives of the dataset in achieving a clean set of tags. Once such a set is reached, tags removed as a result of this or future steps can be salvaged for any useful information that they might contain. Essentially, this step can be used to perform a type of smart sampling on the input dataset, with little possibility of losing important information.

2. **Spelling and Multiwords:** With the open nature of tag spaces, it is difficult to employ dictionaries as ground-truth in spelling correction, which is the common method in text cleaning tasks. Instead, our ground-truth includes the words with the highest *quality* (above a user-chosen threshold), along with any words or tags imported by the user (white-lists). Spell correction is achieved by replacing lower *quality* words that are found sufficiently similar to a ground-truth word with the latter. The threshold on similarity is also decided by the user. Besides fixing spelling errors like "roock" to "rock", this step also unifies different word-forms, such as "visualization" and "visualisation".

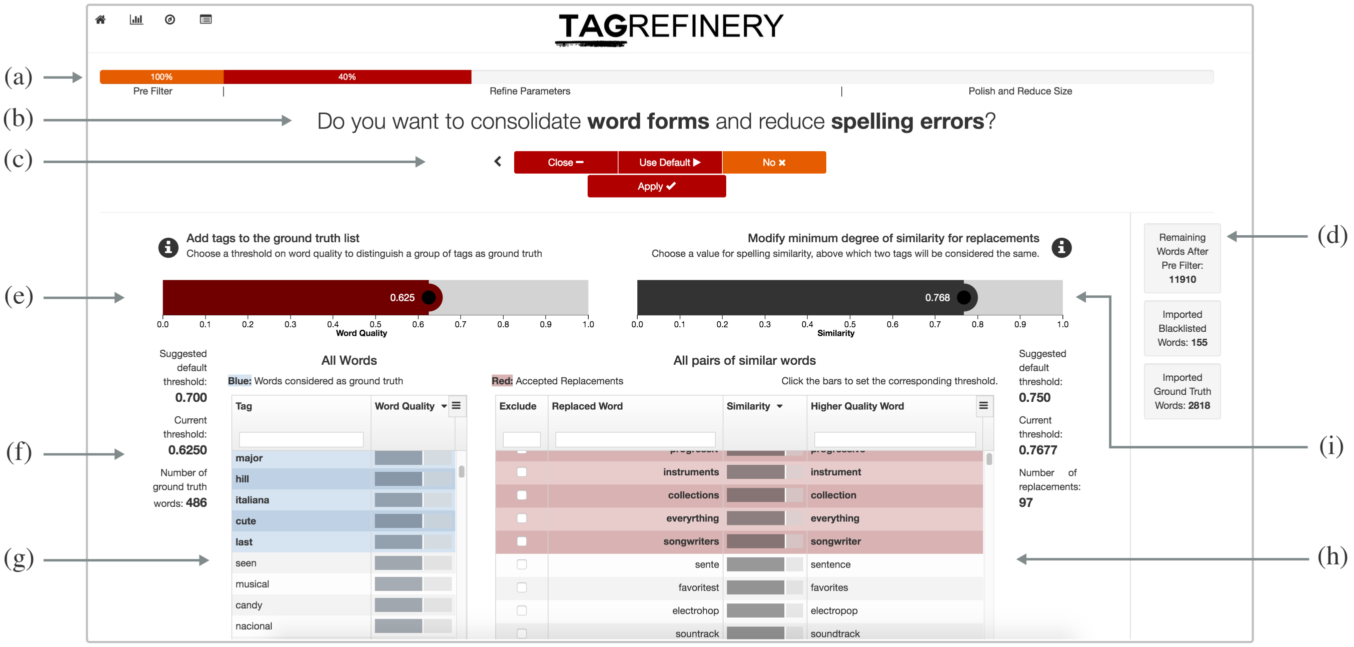
   Once spell correction is performed, we look back at the initial tags and re-construct those that contained multiple words. Two different techniques are used to this end; one looks for frequently appearing sequences regardless of the individual words (such as "hard rock") and one finds unique sequences whose individual words do not appear frequently (such as "Jon Bon Jovi"). We call these two types of tags *frequent* and *unique* multiword tags. This helps us cover a greater span of possible multiword expressions.

3. **Polish and Salvage** Once the multiword tags are reconstructed, the tag *quality* value is re-computed, as the dataset might have changed significantly. At this stage, the user can filter the output set of tags based on tag *quality*, manually remove individual tags (e.g. to remove subjective tags in recommendation use cases) or replace them with others (e.g.

[2] The Million Song Dataset [3]

[3] The data that is fed to TagRefinery is offline and no longer connected to the online source (in this case Last.fm) when processed. If the dataset is updated, our workflow needs to be run again.

[4] *Quality* of single words is the maximum *quality* among all tags that the word originates from.



Do you want to consolidate **word forms** and reduce **spelling errors**?

Add tags to the ground truth list
Choose a threshold on word quality to distinguish a group of tags as ground truth

Modify minimum degree of similarity for replacements
Choose a value for spelling similarity, above which two tags will be considered the same.

Suggested default threshold: **0.700**
Current threshold: **0.6250**
Number of ground truth words: **486**

All Words

Blue: Words considered as ground truth

Tag	Word Quality
major	
hill	
italiana	
cute	
last	
seen	
musical	
candy	
nacional	

All pairs of similar words

Red: Accepted Replacements

Click the bars to set the corresponding threshold.

Exclude	Replaced Word	Similarity	Higher Quality Word
☐	instruments		instrument
☐	collections		collection
☐	everyrthing		everything
☐	songwriters		songwriter
☐	sente		sentence
☐	favoritest		favorites
☐	electrohop		electropop
☐	sountrack		soundtrack

Suggested default threshold: **0.750**
Current threshold: **0.7677**
Number of replacements: **97**

Remaining Words After Pre Filter: **11910**

Imported Blacklisted Words: **155**

Imported Ground Truth Words: **2818**

Figure 2. Spell correction screen from Guided mode. (a) Progress bar for whole workflow. (b) Question asking the user whether the current step should be performed. (c) User's options to skip step, use defaults, or customize parameters of step. (d) Status column showing outcome of each step. (e) Interactive bar for threshold on word *quality*. (f) Statistics on word *quality* threshold and accepted ground-truth words. (g) List of all words and their *quality*. Highlighted words are counted as ground-truth with current threshold on word *quality*. (h) List of all possible word pairs for replacement. Highlighted pairs are accepted with current threshold on word similarity. (i) Interactive bar for threshold on word similarity

to consolidate synonyms), and finally salvage useful information out of tags previously removed in any of the filtering steps. For instance, the term "hiphop" is salvaged from the tag "raphiphopsong" which is likely to be completely lost if no salvaging is performed. This process prioritizes tags with a larger number of words. As an example, if the final set of clean tags includes both "rock" and "hard-rock" and the tag "hardrockfestival" has previously been filtered out from the final set, the part "hard-rock" will be salvaged from it.

We first started by analysing the problems our colleagues had with the Last.fm folksonomy. Through mutliple iterations, the current steps were defined and the order was refined. The most prominent issue in the Last.fm folksonomy (and other social tag spaces) is noise. As such, spell correction is the main language related task performed in the workflow. To be able to fix spelling errors within multiword tags such as "hard roock", we first needed to decompose them into single words. The multiword expression detectors were then added to revive tags that are longer than one word. The earlier versions of the workflow did not include the salvaging step. After a closer look at the results, we observed that numerous meaningful tags could be found within those removed in either Pre or Post-Filtering. This led to the addition of the Information Salvaging step. Naturally, we also required the possibility of quick experimentation with the workflow and real-time interactions with the interface, which is difficult considering the often enormous sizes of folksonomies. This requirement led to the addition of the Pre-Filter step, which acts as a way of sampling the input and increasing performance.

Our envisioned usage of the workflow involves going through Pre-Filter once, then iterating and refining parameters in Spelling and Multiwords, and finally performing Polish and Salvage once to finalize the dataset. This is due to the fact that the middle part of the workflow has the highest impact on the quality of the output tags, and thus is most likely to require iteration and refinement. Figure 1 depicts these three phases and our expected iteration in the middle. A similar diagram [5] is also shown to the users when they open the tool for the first time, in order to help them form the correct mental model.

TAGREFINERY – INTERACTION DESIGN

Pre-design analysis

Users:

The common way of cleaning text data is with the use of various NLP tools that are accessible through command-line interfaces. While users with a background in computer programming may feel at home with such methods, the population of researchers in need of tag cleaning tools is much more diverse. Therefore, our initial goal was to create an easy to use application with no assumption of previous knowledge on NLP or data cleaning techniques.

However, given the range of expertise in our targeted audience, we quickly realized that while novice users may be satisfied with the specific sequence of tasks in our workflow, more advanced users might require to run the tasks out of order and with more granular control. Through several iterations, this requirement finally led to two interfaces modes targeting the two

[5] excluding the decomposition step which is always hidden from the user

ends of the spectrum of expertise: **(a) Guided** mode for the newcomers and novices in data cleaning, and **(b) Advanced** mode for users that are either experts in data processing or have ample prior experience with TagRefinery itself. This mode provides additional parameters which are not visible in the Guided mode.

The parameters from the Spelling and Multiwords steps are highly interconnected. Early experiments with the workflow indicated that often the user would need to access and study these parameters and their relation altogether in one large overview. Therefore, we added a screen called the **Linked** view which allows for quick iterations over different parameter combinations from these steps with immediate feedback from the interface. This mode is intended for all user types.

Effort vs. Quality of output:

Besides supporting different kinds of users, our two interface modes also cover a spectrum between effort put into the process and the quality of the output. The task of cleaning a folksonomy is itself an iterative process. Our colleagues from the music recommendation research [12] went through numerous versions of their tag set as they refined and finalized their requirements for the output. By providing the Guided mode and default parameters, which were empirically found through experimentation with different datasets, TagRefinery allows the user to quickly produce an output without much effort. As the user delves deeper into the parameters, the output can become more and more clean and polished.

Interface modes

The user is first presented with the Guided mode, which walks him/her through the workflow. Once the end is reached, the user is encouraged to try the Advanced mode and the Linked view through prominently placed buttons on the screen. As such, the Guided mode acts as an introduction to the workflow and helps the user understand each step.

Guided mode

In each step of the Guided mode, the user can either choose to skip the step completely, use our default parameters, or customize the parameters. Each step also provides a brief explanation of what it does and how the underlying technique operates. The default settings use our preset thresholds and parameters which have been chosen based on empirical observations. When the user chooses to customize parameters, the view is expanded to display the controls and visualizations for manipulating the parameters. To convey to the user our intended way of using the workflow and having them form the correct mental model, the Guided mode displays a simplified form of the diagram from Figure 1 in the very beginning. In addition, there is a progress bar at the top of the screen which is divided into the same major stages in Figure 1[6].

Figure 2 depicts a screen-shot of the spell correction step of the workflow in the Guided mode, expanded for parameter customization. The question (Figure 2(b)) below the progress bar asks the user whether (s)he wants to perform the step, and the user's options are displayed in the form of three buttons: "Yes", "No" and "Use Default". The "Yes" button also has a gear icon next to it which indicates that clicking on "Yes" would allow the user to customize the parameters of the current step (Figure 2 (e-i)). In the screenshot, the "Yes" option has been chosen, which turns its button into "Close" and adds an "Apply" option (Figure 2(c)). To increase the user's awareness of the current status of the dataset and workflow, throughout the Guided mode, a status column on the right side (Figure 2(d)) displays the outcome of every past step. For instance, it displays how many words remain in the dataset after Pre-Filtering and how many replacements have been made in the spell correction phase. The dark red bar at the top left (Figure 2(e)) can be used to set a threshold on word *quality*, to designate a group of words as ground-truth in spell correction. The statistics on the left (Figure 2(f)) indicate the number of currently selected ground-truth words and the current threshold. The list on the left (Figure 2(g)) shows all words in the dataset along with their *quality* (light grey bars) and the ones that qualify as ground-truth with the current threshold (with blue highlight). The list on the right (Figure 2(h)) shows all pairs of possible replacements between lower *quality* and higher *quality* words and their degree of similarity (light grey bars). Lastly, the dark grey bar at the top right (Figure 2(i)) can be used to specify the required minimum degree of similarity for a replacement to be accepted (red highlight). The thresholds set by the two sliders at the top are linked to the respective lists below. Manipulating each causes the small white vertical lines in the lists to move with the threshold and the lists to scroll with the current borderline word. Both lists can be sorted based on various attributes, and searching for specific words is possible as well.

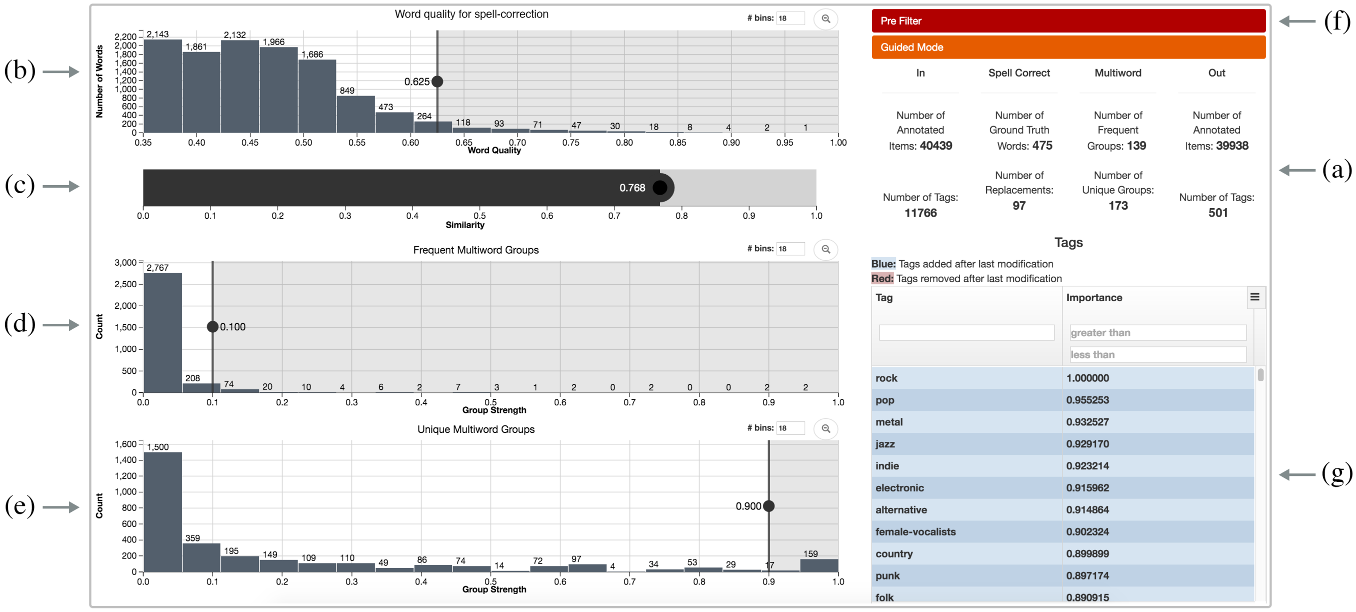
Advanced mode

This mode provides a structural, and quick to navigate interface to all the steps of the workflow and includes access to some additional customizations. For instance, the user can specify a lower bound on word length in spell correction, in order to prevent small words from being replaced. This is supported because the similarity scores between small words can be much higher than between longer words, and this can cause undesirable replacements. Outside the added customizations, most controls and widgets resemble those shown in the Guided mode in order to retain visual consistency.

Linked view

Figure 3 shows the Linked view. The current status of the workflow is reported at the top right (Figure 3(a)). On the left, the histogram for setting the ground-truth threshold in spell correction (Figure 3(b)), the bar for setting the word similarity threshold in spell correction (Figure 3(c)), and histograms for frequent (Figure 3(d)) and unique (Figure 3(e)) multiword tag detection are displayed. The user can also choose to see the pre-filtering histogram by clicking the Pre-Filter button at the top right (Figure 3(f)). The list of all output tags after spell correction and multiword tag detection is also shown in the bottom right (Figure 3(g)). Changing any of the parameters elicits immediate feedback from the interface. The list on the

[6] The names of these high level phases were slightly different in the evaluated interface. They were: "Pre-Filter", "Refine Parameters" and "Polish and Reduce Size". After final evaluations, we learned that they needed to change in order to better reflect the workflow. The updated names are used in Figure 1.



Pre Filter

Guided Mode

In	Spell Correct	Multiword	Out
Number of Annotated Items: **40439**	Number of Ground Truth Words: **475**	Number of Frequent Groups: **139**	Number of Annotated Items: **39938**
Number of Tags: **11766**	Number of Replacements: **97**	Number of Unique Groups: **173**	Number of Tags: **501**

Tags

Blue: Tags added after last modification
Red: Tags removed after last modification

Tag	Importance
	greater than less than
rock	1.000000
pop	0.955253
metal	0.932527
jazz	0.929170
indie	0.923214
electronic	0.915962
alternative	0.914864
female-vocalists	0.902324
country	0.899899
punk	0.897174
folk	0.890915

Figure 3. The Linked view of TagRefinery. (a) statistics (b) word *quality* threshold for spell correction. (c) similarity threshold for spell correction. (d) group strength for frequent multiword tags. (e) group strength for unique multiword tags. (f) show/hide pre-filtering step. (g) list of output tags highlighting changes made in this screen.

bottom right also demonstrates the effect of each change to the parameters by highlighting (in blue and red) tags which were added or removed due to the most recent change. Hence, this screen can be used to quickly study the interactions between parameters and how they affect the final set of tags.

Formative evaluations and design choices

TagRefinery went through major transformations in its course from paper prototypes to web application (Figure 2). Images of some of these old designs are provided in supplementary materials.

During the design process, 4 stages of formative evaluations were performed, each with 3 new users. Out of these 12 participants, one was an NLP professor and 4 others had a general computing science education. The rest came from non IT related backgrounds. Each user had 30 minutes to play around with the tool, and the main focus in these tests was the usability of the Guided mode. Some of the design choices we made through this process are discussed below.

Task flow:

The initial prototypes of the interface were similar to what is currently our Advanced mode. However, we found that due to the unfamiliarity of the workflow for our participants, they found it difficult to understand the logic behind the interface, and the order in which they were supposed to perform the tasks. This is in spite of the interface containing visual cues as to what the correct flow of tasks was. We decided that having a Guided mode that blocks access to later steps was critical in easing the users into the tool. This proved to be a successful strategy in later tests and turned the tool from something that was deemed unusable by most participants to a tool that was easy to understand and follow.

Visualizing the tags:

One of the common ways of visualizing tags is with tag-clouds, however, this technique was not suitable for our purposes as our datasets can contain thousands of tags and a tag-cloud can only provide a summary of the top tags with no real use in inspecting thresholds and outputs of the workflow. Scatter-plots are similarly unsuitable as the extreme long-tailed distribution of tag *quality* or occurrence makes it difficult to discern any meaningful clusters or patterns. Hence, we settled for simple colour-coded interactive lists accompanied by histograms.

Histograms vs. bars for parameter setting:

Initially, all thresholds were set through interacting with histograms (with user adjustable numbers of bins) that showed distributions of the value on which the threshold was being set. During our formative tests we observed that for some tasks the histograms could have adverse effects on the users' decision making. For instance, when selecting a group of words as ground-truth in spell correction (Figure 2), due to the extreme long-tail shape of the histogram of word *quality*, some users' first impression was that the dataset had very low overall quality. On the other hand, some users benefited from the added information and tried to set their threshold at the start of the long tail, which indicates a slow-down in changes to word *quality* beyond that point. This strategy can be sensible for choosing high *quality* words as ground-truth because it detects a major shift in word *quality* and removes words beneath it. It however does not lend well to choosing the threshold on similarity between pairs of words for a replacement to take place. That is because for similarity, one would generally look for higher values (75% or higher) to be conservative about losing information in the replaced words. However, the histogram for similarity had a shape that was analogous to that of the word *quality* histogram, and this could invoke similar visual strategies in the user for the two tasks. In the current version, the spell correction step in the Guided mode does not show histograms (still shown in Linked view) to prevent the above issues, however, finding a visualization that helps in choosing the similarity threshold remains an open design question.

Awareness of dataset and workflow state:
Some of our formative testers tended to forget about the changes they had made to the dataset in earlier steps. For instance, a user had filtered all the words of the dataset in the Pre-Filter stage and was wondering why he was getting empty lists of words later on. The status column on the right side in the Guided mode (Figure 2(d)) was added to rectify this issue by always providing the user with contextual information on the state of the dataset up to the present step. Clicking on each of the blocks in this status column takes the user to the corresponding step.

Observing and interacting with lists of tags and words:
In our earlier designs we downplayed the task of inspecting the lists of words in all steps by making them only appear on-demand. The reasoning behind this decision was that we found the lists to add to the visual clutter of the tool. In our usability tests however we observed that most participants were interested in checking the lists, paid extra attention to how their interactions changed the outcome of each step, and tried to obtain this understanding through constantly switching between parameter setting and scrolling the lists. Therefore, we made the lists a prominent part of all screens and implemented links between them and parameter controls to give the user a constant feedback.

Refining interacting parameters:
Our spell correction algorithm (Figure 2) deals with two parameters that are closely intertwined: the threshold on word *quality* that decides what words should be treated as ground-truth, and the threshold on similarity between pairs of words to accept a replacement. Designing ways of changing the two parameters while staying aware of the changes happening to the dataset proved difficult. Early on, the presentation was different; the user had to click on each ground-truth word on the left side to view a cluster of similar words that would be replaced with it on the right side. We also allowed the user to set different thresholds on pairwise word similarity for each of these clusters of similar words, which gave the user much more control over the spell correction task. However, both of these proved too complex and often unnecessary. It was more important to show an overview of all the replacements than to provide granular control over each cluster. As such, the list on the right side was changed to the current list of pairs of words, and the threshold was limited to one global value. The possibility to reject or add individual pairs of words to replacement was added to rectify the loss of control that this new design caused. Earlier designs also showed numbers for word *quality* and similarity strength, instead of the light grey bars and visual thresholds, and this did not end up successful either. In the final version we visualized these values and made the lists interact with the threshold setting sliders at the top.

Details on demand:
As Shneiderman's mantra points out [23], details should be provided on demand. This was crucial in the design process of the Guided mode. Each step asks a question from the user and provides a brief high-level description of the underlying algorithm, but hides all other distracting elements in order to help the user focus on the question. All of our users liked the fact that they could choose the "Use Default" button for each step and get familiar with the workflow without worrying about the underlying mechanisms. An additional level of detail is shown when the user hovers the mouse pointer over a boldface phrase or word in the question or the brief description of the algorithm. For instance, the phrase "word forms" is written in boldface in the spell correction screen, hovering the mouse over which opens a tool-tip that says: "Like - favourite, favorite, favourites or favorites". A third level of detail is provided when the user clicks on the "i" icons on the screens (e.g. in Figure 2, above the interactive threshold bars). This brings up a text with additional information on what the corresponding widget does.

SUMMATIVE EVALUATIONS

Summative evaluations were done on both usability and usefulness of the final design. None of the users had previously seen the tool in a formative test.

Usability

This was tested with 6 participants (3 females, median age = 26). One was a data scientist and the rest had a general computer science background. Each session took 60 minutes, which was divided into 10 minutes for introduction to the interface and brief training, 30 minutes for the main task, and 20 minutes for an interview and the SUS questionnaire [5]. In the main task, users started with the Guided mode and were given a 3000-word subset of the Last.fm dataset (to have fast run-times). The task was to clean the dataset and output 200 tags for a recommendation service. Once the users reached the end in the Guided mode, they could progress to the Advanced mode or the Linked view. All sessions were screen captured and the users were encouraged to "think-aloud" while performing the task. It is worth noting that this was solely an attempt at testing the usability of the interface as a perfect output needs much more iteration and fine tuning than what a lab study allows. As such, we did not provide any specifications for the output and the users' results were not comparable to each other or any reference cleaned dataset.

All users finished the main task. The average SUS score was 80.83 with a standard deviation of 5.16. This is considered a strong score in the literature[7] [1] and indicates a positive response from the participants.

The interviews were manually coded to extract the most prominent concepts and concerns. All 6 users mentioned that they had never seen a tool with similar capabilities before. All participants also appreciated the sequential nature of the Guided mode and its ease of use, remarking that the possibility of choosing the "Use Default" option gave them peace of mind over their decisions if they were not sure of what values to set

[7] Literature average is said to be around 70 [1]

for the parameters. Four participants reported that they liked the various lists of tags and how they gave them feedback about the current state of the system. Another prominent concept (4 users) was the clean design which let the users focus on one problem at a time without confusing them with unnecessary clutter. Two users found the Linked view most interesting and liked its capabilities to "reason about the connection between the parameters and the output". On the negative side, 3 users reported that not all steps were completely clear for them, 2 said that the spell correction screen was difficult to understand at first use, and 1 user did not initially comprehend what was meant by a "multiword" tag.

Usefulness

Here, we first reflect on our results from the Last.fm dataset and the feedback we got from our collaboration with the music recommendation project [12]. We then report on our case studies with 2 expert participants who were given 1 hour to experiment on their own data using TagRefinery while giving qualitative feedback.

Case study 1: Song tags from Last.fm

With this dataset, the overarching goal was to reduce the size of the set of tags in a way that the outcome was both descriptive of the music collection (not losing important tags), and at the same time small enough so that a user of the music recommendation tool could look through all tags in a short time. Initially, the dataset contained $428,887$ songs (with at least one tag), $250,898$ tags, and $5,476,264$ (song,tag) annotations. In the end, our colleagues settled with 358 final tags after manual polishing (422 before polishing). With these, the number of songs with at least one clean tag was reduced to $226,612$, and the number of (song,tag) annotations was reduced to $2,299,154$.

The value of the information salvaging step was clearly visible in the Last.fm dataset. Without salvaging, the final dataset with 358 tags contained $1,748,002$ (song,tag) annotations. However, after salvaging useful information from removed tags, this number rose to $2,299,154$. This translates to a significant increase of 31%.

As discussed earlier, it is difficult to conceive a holistic measure for the quality of the final tag set, however, a close inspection of the resulting tags revealed a desirable level of overall quality in them (before manual polishing). There were very few tags for which we could not find a meaning that was related to music description through searching the web. These were manually removed in the polishing step. Our music recommendation colleagues also required all subjective tags (like "awesome") to be removed, which was done manually.

In general, the feedback from both our colleagues and the users of their recommendation tool has been highly encouraging. There are however areas that could use improvement, such as handling synonyms and antonyms for item retrieval purposes.

Case study 2: Research paper keywords

One of our expert participants' research involved cleaning keywords from the visualization literature. He had a dataset of $5,369$ keywords which were made up of $2,656$ unique single words and $8,211$ (paper, keyword) pairs. Like other users, he mentioned that he did not know of a tool with comparable capabilities and was intrigued by the overall idea. That said, he found some issues at first use for which the tool was not configured properly, and required some missing features.

For instance, the user was surprised to see some of the detected multiword tags that seemed inappropriate. One example was the multiword tag "h.5.2 information interfaces", which had the sixth highest occurrence frequency in his dataset. After some exploration, he realized that since a large number of keywords contained strings such as "h.5.2 [information interfaces and presentation]:", the initial parts of these keywords had been identified as one multiword tag. To fix this, we had to modify our workflow to look for sequences of tags longer than 3 words, which is our default when working with folksonomies. Furthermore, removing special characters like "]" and "[", which the tool performs by default, proved detrimental in the case of paper keyword data. However, since the tool gives the user control of what characters to keep or remove in the Black-Listing step (from the Spelling and Multiwords stage in Figure 1), this issue can easily be fixed.

This user also felt a need for specifying custom regular expressions that would allow him to remove certain types of undesirable tags instead of going through the lists manually. In addition, he expressed a need for more contextual information on demand, like being able to quickly observe the context (keyword) from which a word or multiword tag had been extracted, and having this integrated into the spell correction and multiword tag detection steps. Currently, this functionality is only available in the final screen.

Case study 3: Paper abstracts

The data and goals here were fairly different from our intended use cases of TagRefinery, however, we still found the results interesting. The user for this case study had full abstracts of academic papers, which he needed to clean before analysing for topic models. His dataset consisted of 4,037 abstracts, which contained 16,471 unique words. Each abstract was fed to TagRefinery as a single tag containing hundreds of words. The goal here was to consolidate different word-forms to one (such as "visualization" and "visualisation"), without applying incorrect replacements. As such, the approach here was more conservative than what we typically have with folksonomies.

This participant also found TagRefinery simple and straightforward to use on his data. While he explored his words, he realized that he needed to remove all words that contained no alphabetical characters, such as 0/, 017, 043, etc. Like the previous case study, this again calls for the ability to remove strings that match user provided regular expressions. This expert also needed to filter out words that appeared in all of the abstracts. He achieved this using the histogram in Pre-Filtering which shows the occurrence distribution of words. Another interesting phenomenon was that using our tool, the user realized that there were types of words that he needed to remove that he had not realized beforehand. As such, beside supporting his goals, the tool helped him refine his requirements for a clean dataset by exploring his data from a new point of view.

DISCUSSION

Our main goal in this project was to create both a workflow for cleaning social tag datasets, and a usable and intuitive interface that made it easy to interact with the workflow for users with no technical background in text processing or computer programming. Thus, we decided not to evaluate TagRefinery against command-line tools such as Python libraries, which are the usual method for processing text data. Nevertheless, even in the command-line space, no specific framework exists that caters to social tags. None of the 18 users that tested our tool in its various phases of design and development knew of any comparable software. It is evident that there is a gap in the literature when it comes to comprehensive workflows and interfaces for cleaning tag data, and TagRefinery is an important step toward filling this gap.

At its current form, TagRefinery is tailored to cleaning social tag spaces like that of Last.fm; a fact that is confirmed by the positive feedback from our colleagues who worked on a music recommendation tool. Nevertheless, we were also curious to know how well the tool could be expanded to other types of data, such as keywords of academic papers. When working with users who brought their own data, we were delighted to find that TagRefinery could help them explore their data effortlessly and reason about its characteristics. Using the tool gave these participants a level of insight into the distributions and patterns of their words and multiword tags that they did not possess previously, helping them form new requirements for their cleaned datasets.

The reaction to our interface modes was also encouraging. The participants who did not have much experience in text processing appreciated the Guided mode and the trust it induced. On the other hand, participants with more background knowledge tended to like the Advanced mode more, as it provided them with precise and quick control over all aspects of the workflow. Along with valuable feedback on our final design, our summative evaluations also helped us gain more insight into how these two types of users interacted with TagRefinery and how their needs could be better served. These matters are discussed below.

Design lessons and remaining questions

Guided mode as gateway:

As we had hoped, the Guided mode proved useful in conveying the correct mental model to the users. It rarely happened that a participant would choose to skip a step. The prominent choices were to either trust the system by choosing the default option, or expanding the task to observe it more closely. This indicates that the interface succeeded in making the users care about all the steps of the workflow. Even for our more advanced users, we found the Guided mode a necessary introduction and gateway to the workflow. There were however some interesting differences in how users interacted with the interface, which are discussed next.

Novice vs. expert/Quick vs. meticulous:

Using the application for the first time, participants with no data processing experience tended to start by quickly going through all the steps and pressing the "Use Default" button, in order to see what the results looked like. They then went back and closely inspected each step. This way, the Guided mode provided them with a clear starting point and helped them warm-up to the workflow. This is while users with prior experience in data cleaning meticulously went through each step by expanding its customization controls and trying to reason about what the algorithms did before progressing to the next step. This observation emphasizes the importance of strong default parameters, as these are the main points of entry for novice users and build their trust in the tool. These are currently chosen based on experimentation with a number of folksonomies, but there is much room for improvement. A worthwhile effort is to build algorithms that examine the dataset and the possible outputs for each threshold in order to achieve smart suggestions based on the data.

On the other hand, our expert users required some functionalities that were not included in TagRefinery, such as the regular expression feature discussed before. These users wanted to pick and choose tasks and possibly add their own algorithms in-between. Therefore, one important future improvement for our Advanced interface mode is to adopt a module-based paradigm, where the users can build their own widgets into the interface or export and import data between steps.

Refining interacting parameters:

Earlier we explained how we reached the current design for the spell correction screen which deals with two interacting parameters (Figure 2). This design made the task significantly more understandable to our final participants. The white vertical threshold line overlaying the lists made it clear as to how exactly the parameters were acting on the dataset, and the real-time highlighting of accepted and rejected replacements while changing the thresholds made the interactions between the parameters more transparent.

That said, the problem of interacting parameters is not limited to spell correction. The whole workflow can be seen as a system of interacting thresholds with one output. Continuing from the above point on smarter default parameters, coming up with a holistic quality measure for any given output can be another worthwhile addition. Based on such a measure, the output space can be sampled based on various values for the interacting parameters and this can greatly reduce the amount of time the user needs to dedicate to examining and refining them. A holistic quality value can be achieved through combining our current language independent techniques with domain specific dictionaries (music, paper keywords, etc.) and web mining approaches (for words that do not appear in any dictionary). As the quality of the output can be subjective and dependent on the specific use case, the user should be allowed to give weights to different quality factors in achieving the final measure.

Explanations, terms, and expressions:

In order to cater to all users and not just data cleaning or NLP experts, one critical factor in designing the interface was coming up with terms and phrases that would clearly describe what each step or widget performs. The names of steps and various parameters have gone through numerous iterations, and the current versions have proved relatively successful in usability tests. The short explanations provided in each step (before the view is expanded for parameter customization) were carefully worded to quickly give the user an understanding of the logic behind each step. In our final evaluations we saw that most novice users (who quickly went through the steps on first use) ignored these the first time they went through the workflow, but when they returned and studied each step, they found that the explanations helped them gain a better understanding of the algorithms.

Limitations and future work

We have achieved an initial understanding of how our tool might be used for various text processing tasks, however, our user base for summative evaluations was fairly small. The next step would be to make the tool publicly available in order to obtain more real world feedback. We have already started this phase and are in the process of collaborating with more researchers.

One shortcoming of the current approach is that new words cannot be quickly added to the final dataset as that would require running most of the workflow again. Considering the highly dynamic nature of social tag spaces, this is important to address for use cases that require up-to-date datasets. One way of alleviating this issue can be a supervised learning system which extracts the “rules” of a specific dataset, so that such rules can easily be applied to newly inserted tags in order to classify them as either truth or noise. Models that are widely used in speech recognition and spelling correction can be the place to start. Examples include Hidden Markov Models (HMM), Naive Bayes and Maximum Entropy classifiers.

Regarding the workflow, incorporating algorithms for removal of other types of noise would be an important next step. For instance, phenomenons such synonymy and polysemy[8] are not handled by TagRefinery and should be rectified in the Manual Polishing step by the user. Another area of improvement can be providing a choice between different tag *quality* formulas. For the Last.fm data, we use the available meta-data on popularity of songs and the weight of a (song,tag) annotation to compute tag *quality* prior to the workflow. Building visual tools for the user to help define this measure based on their dataset and its available meta-information is a worthwhile design space to explore.

Our case studies with expert users helped us better understand what functionalities are most important when it comes to adapting TagRefinery to data types other than social tags. One prominent requested feature was the ability to specify custom regular expressions at various stages of the workflow in order to remove certain types of tags. This functionality is critical when one deals with a tag space that is more structured than a typical folksonomy (such as paper keywords), as recurring patterns of characters and terms are more likely to occur in such cases. In addition, compared to cleaning folksonomies, some text processing tasks are more conservative about keeping the words and tags intact and making only the most trusted replacements in spell correction. To adapt to these requirements, the workflow would have to be able to switch to different sets of default parameters and blacklists (e.g. removing or keeping special characters in the beginning).

Moving beyond various bundles of default values, as discussed earlier, the tool can be transformed in more radical ways to accommodate different workflows. As it stands now, some of the steps in the workflow cater well to other text processing tasks (e.g. spell correction and multiword detection), but some might only be needed in a folksonomy cleaning scenario (e.g. information salvaging). Having a number of preset or modular workflows that target various tasks can be a step in the right direction here.

CONCLUSION

TagRefinery was built to give researchers who work with open tag spaces (folksonomies) an easy to use workflow and visual interface for cleaning tags and (item,tag) annotations. Both the workflow and interface were iteratively designed in collaboration with a music recommendation project based on Last.fm tags [12], and were refined through several formative usability tests.

TagRefinery targets a range of user expertise from novice to advanced and provides two different interface modes to accommodate this spectrum: Guided, and Advanced. The Guided mode moves the users through all the steps of the workflow one by one, giving them the ability to use default parameters, customize parameters, or completely skip the step. The Advanced mode provides a more complex interface with direct fine grained control over all parameters of the workflow. In addition, TagRefinery includes an overview screen called the Linked view, which provides a big-picture view of the most important parameters of the workflow along with immediate feedback. At all stages, the interface provides visual and statistical aids to the users to help them customize the parameters of each step according to their dataset and its intended use.

The principal stages of our workflow are (a) Pre-Filtering, (b) Spelling and Multiwords, and (c) Polish and Salvage. To remove language dependence and keep up with the dynamic nature of folksonomies, our algorithms are all based on statistical methods, with the possibility of adding user provided dictionaries for increased accuracy.

Our user evaluations showed a high level of usability, with even non-expert users easily understanding the logic and mental model of the workflow. Our expert users, on the other hand, were enthusiastic about the fact that they knew no other tool with similar functionalities. They found that TagRefinery gave them novel ways of exploring and inspecting their data, giving them unexpected and sometimes surprising insights.

ACKNOWLEDGEMENTS

We would like to sincerely thank Michael Sedlmair for his valuable input during various stages of this project. We would also like to thank the participants of our user studies for their constructive feedback, and the reviewers of this paper for their comprehensive comments during the revision cycle.

REFERENCES

1. Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding

[8] One word having different meanings, such as “Love” meaning both a user liking a song and a song being about love

an Adjective Rating Scale. *J. Usability Studies* 4, 3 (May 2009), 114–123. http://dl.acm.org/citation.cfm?id=2835587.2835589

2. Dominikus Baur, Jennifer Bttgen, and Andreas Butz. 2012. Listening Factors: A Large-Scale Principal Components Analysis of Long-Term Music Listening Histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. 1273–1276. http://dl.acm.org/citation.cfm?id=2208581

3. Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR'11)*. International Society for Music Information Retrieval, Miami, Florida, USA, 591–596. http://ismir2011.ismir.net/papers/OS6-1.pdf

4. Eric Brill and Robert C. Moore. 2000. An Improved Error Model for Noisy Channel Spelling Correction. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL'00)*. Stroudsburg, PA, USA, 286–293. DOI: http://dx.doi.org/10.3115/1075218.1075255

5. John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7. http://hell.meiert.org/core/pdf/sus.pdf

6. Stefan Evert and Brigitte Krenn. 2005. Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language* 19, 4 (2005), 450–466. DOI: http://dx.doi.org/10.1016/j.csl.2005.02.005

7. Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised Type and Token Identification of Idiomatic Expressions. *Computational Linguistics* 35, May 2008 (2009), 61–103. DOI: http://dx.doi.org/10.1162/coli.08-010-R1-07-048

8. Davide Fossati and Barbara Di Eugenio. 2008. I saw TREE trees in the park: How to Correct Real-Word Spelling Mistakes. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)* (28-30). Marrakech, Morocco, 896–901. http://www.lrec-conf.org/proceedings/lrec2008/

9. Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2012. Parsing Models for Identifying Multiword Expressions. *Computational Linguistics* 39, 1 (Nov 2012), 195–227. DOI: http://dx.doi.org/10.1162/COLI_a_00139

10. Manish Gupta, Rui Li, Zhijun Yin, and Jiawei Han. 2010. Survey on social tagging techniques. *ACM SigKDD Explorations Newsletter* 12, 1 (2010), 58–72. DOI: http://dx.doi.org/10.1145/1882471.1882480

11. Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering* 11, December 2003 (2005), 87–111. DOI: http://dx.doi.org/10.1017/S1351324904003560

12. Mohsen Kamalzadeh, Christoph Kralj, Torsten Möller, and Michael Sedlmair. 2016. TagFlip: Active Mobile Music Discovery with Social Tags. In *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI'16)*. ACM Press, 19–30. DOI: http://dx.doi.org/10.1145/2856767.2856780

13. Yvonne Kammerer, Rowan Nairn, Peter Pirolli, and Ed H. Chi. 2009. Signpost from the Masses: Learning Effects in an Exploratory Social Tag Search Browser. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. New York, NY, USA, 625–634. DOI: http://dx.doi.org/10.1145/1518701.1518797

14. Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'011)*. ACM, New York, NY, USA, 3363–3372. DOI: http://dx.doi.org/10.1145/1978942.1979444

15. Paul Lamere. 2008. Social Tagging and Music Information Retrieval. *Journal of New Music Research* 37, 2 (2008), 101–114. http://www.tandfonline.com/doi/abs/10.1080/09298210802479284

16. C. Laurier, M. Sordo, Joan Serrà, and Perfecto Herrera. 2009. Music Mood Representations from Social Tags. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR'09)*. Kobe, Japan, 381–386. http://mtg.upf.edu/node/1466

17. Mark Levy and Mark B. Sandler. 2007. A Semantic Space for Music Derived from Social Tags. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR'07)*. Vienna, Austria, 411–416. http://ismir2007.ismir.net/proceedings/ISMIR2007_p411_levy.pdf

18. Emanuele Quintarelli. 2005. Folksonomies: Power to the People. (2005). http://www-dimat.unipv.it/biblio/isko/doc/folksonomies.htm

19. Carlos Ramisch, Helena De Medeiros Caseli, Aline Villavicencio, Andŕe Machado, and Maria José Finatto. 2010. A hybrid approach for multiword expression identification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6001 LNAI (2010), 65–74. DOI: http://dx.doi.org/10.1007/978-3-642-12320-7_9

20. Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2006. Orthographic Errors in Web Pages: Toward Cleaner Web Corpora. *Computational Linguistics* 3, December 2005 (2006), 295–340. DOI: http://dx.doi.org/10.1162/coli.2006.32.3.295

21. Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. *Multiword Expressions: A Pain in the Neck for NLP*. 1–15. DOI: http://dx.doi.org/10.1007/3-540-45715-1_1

22. Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *Transactions of the Association for Computational Linguistics* 2 (April 2014), 193–206. http://www.transacl.org/wp-content/uploads/2014/04/51.pdf

23. Ben Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages (VL'96)*. 336–343. http://dl.acm.org/citation.cfm?id=832277.834354

24. Louise F. Spiteri. 2007. The structure and form of folksonomy tags: The road to the public library catalog. *Information Technology and Libraries* 26, 3 (2007), 13. http://ejournals.bc.edu/ojs/index.php/ital/article/view/3272

25. Kristina Toutanova and Robert C. Moore. 2002. Pronunciation Modeling for Improved Spelling Correction. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL'02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 144–151. DOI: http://dx.doi.org/10.3115/1073083.1073109

26. Tim Van de Cruys and Begoña Villada Moirón. 2007. Semantics-based Multiword Expression Extraction. In *Proceedings of the Workshop on a Broader Perspective on Multiword Expressions (MWE'07)*. 25–32. http://dl.acm.org/citation.cfm?id=1613704.1613708

27. Jesse Vig, Shilad Sen, and John Riedl. 2011. Navigating the tag genome. In *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI'11)*. 93–102. http://doi.acm.org/10.1145/1502650.1502661

28. Ju-Chiang Wang, Yu-Chin Shih, Meng-Sung Wu, Hsin-Min Wang, and Shyh-Kang Jeng. 2011. Colorizing Tags in Tag Cloud: A Novel Query-by-Tag Music Search System. In *Proceedings of the ACM Conference on Multimedia (MM'11)*. DOI: http://dx.doi.org/10.1145/2072298.2072337