Visualization Project: Tag Refinery

Christoph Kralj*

University of Vienna

1 Motivation

In today's world, tags are everywhere. Movies, music, papers and many more things have tags attached. Those should help us find what we want or help us finding new interesting things. I distinguish between lectured and social tags. Lectured are added by experts in the given field and usually of high quality but very reduced and maybe biased. Social tags on the other hand are very diverse and rich on information. This comes with the drawbacks of: spelling error, context specific language, non-informative words and many more problems. Tag Refinery focuses on social tags and tackles there problems by providing a visual interface for the underlying data wrangling algorithm. The final goal is a cleaned and reduced tag space in respect to the user needs.

1.1 Tasks

The main target is a visual interface which let the user easily manipulate the cleaning pipeline and gives instant feedback about the changes. I try to achieve this by solve the following tasks:

Get an overview of the tags Exploring the tag space (Filtering, zooming) Set the parameters of the underlying algorithm (Interaction) Reflecting over the changes with help of graphs (Brushing & linking)

1.2 Users

The user is every person which likes to make use of social tags. An example would be creating a music recommendation app based on emotions. In this case the user can clean the tag space in a way that emotional words are improved.

1.3 Algorithm

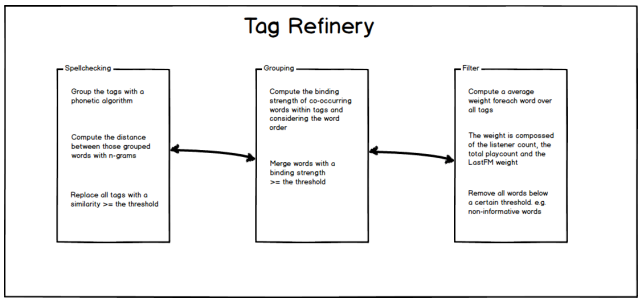

Figure 1: Algorithm overview

The underlying algorithm is part of my master thesis. The overall concept is shown in Figure 1. It consists of three modules: Spellchecking, grouping and filtering. The spellchecking module groups the tags with help of a phonetic algorithm and replaces within each group all words above a given similarity with the highest listeners count word in that group. Grouping is achieved by finding co-occurring words and grouping them in respect to their relative frequency. The last module computes an importance score for each tag by using a weighted-average mean of the Playcount, Listeners and the LastFMWeight. This importance score is used to filter non-informative tags by removing all tags below a certain threshold.

*e-mail: christoph.kralj@gmail.com

1.4 Data

I use music tags from a subset of the million songs database[1] as example dataset. The tags are mined from LastFM[2]. The format of the csv file looks like this:

SongID,SongName,Listeners,Playcount,TagID,TagName,TagWeight

2 Approach

Figure 2: Final interface

Figure 2 shows the final interface. On the top left corner is the scatter plot (x-axis: occurrences; y-axis: importance score) which let the user explore the tag space by using the two filter sliders on the top right corner. The two histograms on the bottom side show the distribution of the the tag occurrences and the importance score. On the right side below the two filter sliders is the control element for the spellchecking module. The user chooses at which percent similarity words will be replaced. Directly below the element is a graph with two bars which show the total replacements in respect to the total tag count. This should give the user a hint how strong the effect of the chosen parameter is. Below the bars is the next input element, this time for grouping. The user can select a binding-strength threshold which is used by the algorithm to decide which words should be grouped. A lower values means more words will be grouped. This can be seen in the list below that element which shows all groups and the corresponding binding-strength. In the lower right corner are the input elements for the importance calculation. Each input is used as weight for the weighted-average computation.

[1] http://labrosa.ee.columbia.edu/millionsong/

[2] http://www.last.fm/

2.1 Design decisions

2.1.1 Overall design

overall design: The scatterplot is the main view and gets most of the space. Below are the small histograms which give the user a overview of the tag space and on the right side are all the controls and inputs. In one of the classes i got feedback for moving the parameter specific graphs to the corresponding inputs. This makes the interface clearer and the impact of the parameters better visible.

2.1.2 Scatter plot

scatter plot: I started with a word cloud which looked really neat and the average user feedback was great. After feedback from a class I realised that the word cloud looks good but does not scale good and has other visual drawback. Thats the reason I switched to the scatterplot which shows the same in a better way.

2.1.3 Group-list

group-list: At the beginning I started with a heat map for showing all groups. This like in the previous example looks nice but did not survive the first user contact. The overall feedback was that its unclear how to interpret the heat map. I used a simpler way of visualizing groups, a list.

2.1.4 Histograms

histograms: The bars of the histogram where in the first prototype vertically aligned with bin size three. I changed this after user feedback to horizontal alignment which fits better into the whole interface and a bin size of one. A bin size bigger than one makes the filtering step unclear, e.g. at which bin are how many points.

3 Implementation

The algorithm is written in java and uses a MariaDB database. I build the interface in Tableau as dashboard. The connection between algorithm and dashboard is realised by csv files. The hardest problem which I did not resolve was the interface between Tableau and Java. The algorithm is not written in a server-client structure and changing would have taken too much time. Aside from this, Tableau has also its limitations. The first is data-joining or data-blending. Using 5 cvs source files with different formats, showed some limitations. The biggest was the impossibility to create a global filter. Even blending two sources to use both in a sheet did not always work. Thats the reason brushing and linking does not work, I was not possible to link the sources properly. A second thing which is not perfect is how you can arrange sheets on the dashboard. There are no ways of creating lines or other structural elements. I still really like Tableau and see its use more in the data exploration and prototyping area indent of creating enduser tools. For this applications is Tableau awesome.

4 Results

4.1 User scenario

Paul (the user) wants to create a new and awesome app for music recommendation. He decides to use social music tags from lastfm. After mining those he realizes that there is a lot of noise in the data. He connects his database with Tag Refinery loads the data (The current prototype uses constant cvs files as database). From this point Paul start to explore the given tag space. Figure 3 highlights the used parts of the interface. With help of the sliders he can explore the tag space which is shown in the scatter plot. On mouseover he gets additional information about a given tag and the histograms below give a overview of the whole space. At this point he might have found misspelled words and words which should be grouped. The next step is the adjustment step. The highlighted parts in Figure 4 give him the opportunity to adapt the algorithm. After changing the parameters he can reflect what happened in the in Figure 5

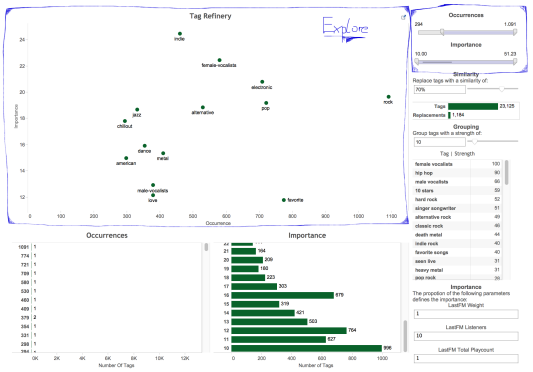

Figure 3: The exploration parts of the interface are highlighted in blue.

Figure 4: The adjustable parameters are highlighted in red.

highlighted areas. Maybe after reducing the similarity too far, Paul realises there are too many replacements and only very few tags are left. Other things which might be interesting are the distributions of the tags after adjusting the parameters. In Fig 5 i a really high peak of tags at an importance score of 50. After exploring this tags in the scatter plot he can see that this tags have nothing to do with emotion. This leads to another iteration in the workflow (Explore - Adjust - Reflect) until the point he is satisfied with the tags.

4.2 Performance

It works smooth with 20000 tags but I am not sure if Tableau can handle half a million tags or more. I tried once to open a file with 3 million rows and Tableau crashed each time. I think in a D3 - Server design the algorithmic code will be the bottle neck and needs to be as fast as possible.

4.3 Feedback

My sources are two non computer scientist which looked gave me feedback at different steps of the interface creation process and the class feedback. The class feedback was really helpful and was the reason I removed the word cloud and changed the overall layout. The feedback from the non computer scientist was really interesting. They showed me that complex or unusual graphs make it really hard to understand what happens. In addition they highlighted the importance of a clear and good naming scheme for the parameters and graphs.

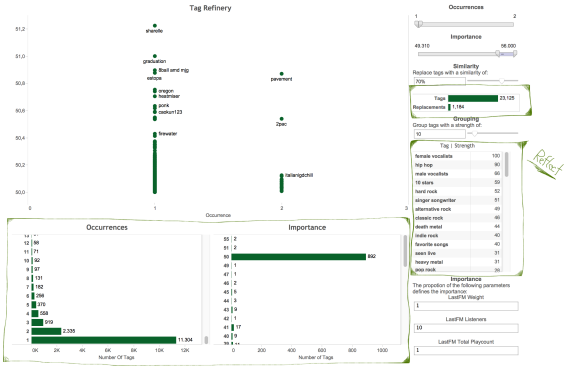

Figure 5: The graphs for reflection are highlighted in green.

5 Discussion

The overall interface works in a nice and smooth way. I have removed everything which I think is not necessary for finding the best parameter combination for a given problem. Still there are a few points which can be improved. Especially the scatterplot which is good but not perfect may be a good starting point. Here are a list of strengths and weaknesses of the interface:

Strengths:

- Tag Refinery does not distract the user from his tasks by providing a simple and easy to use interface.
- It gives a good overview of the tag space and lets the user explore it interactively.
- This in combination helps the user find his optimal parameter combination.

Weaknesses:

- The scatterplot gets cluttered in the lower left corner.
- Importance calculation is hard to understand.
- All graphs show only overview.

5.1 Lessons learned

I learned a lot about how important it is how you name things to reduce ambiguity. Also I did not know the idea of mockups and they helped me a lot designing the first interface and getting my ideas onto paper. From this point I realised that fancy visualization techniques (heatmap, word cloud) should be used with care and at the end may be a simple visualization the better choice. From the technical side I am not aware of any better way of creating a visual prototype than Tableau. But on the other hand I think its not suitable for creating a end-user app. I think a good way of creating a visual tool is to create a prototype interface in tableau and implementing it afterwards in something like D3.

Acknowledgements

I wish to thank Torsten Mller, Mohesen Kamalza, the Vis class and my non-computer scientist users.