Processing Last.fm Tag Data

We used the complete Million Song Dataset as basis for our tag processing. We mined all the tags of all songs from Last.fm in February 2015, using the Last.fm API. At this point, the database consisted of 416,212 songs and 529,088 distinct tags. Due to the fact that our final lab study was performed in *anonymous country*, and that we played audio from Spotify, we filtered the library to songs available from Spotify in this country, leaving us with 333,857 songs and 482,325 tags.

In order to improve the tag quality, we had to perform multiple data wrangling steps. (1) We first removed special characters (- _ : ; /), stop-words, and prepositions, and changed all tags to lower case.

(2) Next, we computed an *importance score* for each tag to determine whether it should remain in our dataset. For this score, we used the weights of each tag/song association provided by Last.fm, along with the number of listeners and play counts for the songs. Since the distributions of the two latter numbers resembled a logarithmic one, we used logarithms in our formula, to bring them back into a more linear space. The formula for *importance score,* normalized to output a value between 0 and 1 for each tag was:

$$\frac{1}{\max\_\text{over\_all\_tags}(\sum_{all\_songs\_with\_tag} lastfm\_weight(\log(listeners) + \log(playcount))}$$

As such, each occurrence of a tag contributed to its final importance score and this contribution was scaled based on the popularity of the songs that the tag occurred in and the weight of association provided by Last.fm for each occurrence.

(3) We next used the Double Metaphone similarity replacement algorithm to find misspelled words and replace them with close words that had the highest importance scores. This algorithm uses the phonetic codes of words to group them. Within all groups, it then computes the Jaccard index between all words. Low importance tags with small distances to a high importance tag are then replaced with the latter. Since there is no ground truth for Last.fm tags and the "wisdom of the crowd" is the decisive factor, we did not use any dictionaries in this step. Use of dictionaries could force the algorithm to choose a "correct" but unpopular tag in place of a popular one that the community uses.

(4) Our fourth step was to detect compound tags like "hard-rock". We used a two stage process for this purpose. In one stage we computed the relative frequency of all 2 and 3-word grams and grouped together all the word grams with a frequency above a certain threshold. We then computed Jaccard distances between all groups and once more put all words with distances below a certain threshold in a new group.

(5) At this step, we ran the similarity replacement algorithm from step 3 again, in order to find misspelled groups.

(6) Since steps three to five change the tag space, in this phase, we computed importance scores for all tags again.

(7) Next, we took the top 550 tags with the highest importance scores and manually went through them to remove subjective terms like “awesome” and “favourite”.

(8) With the help of a regular expression matcher, we then searched for important tags inside unimportant tags to salvage any usable information. This greatly increased our information about the songs (by a factor of 30%). An example outcome is extracting the tags “hip-hop” and “rap” from “raphiphopsong”.

(9) In the final step, we computed importance scores once again for the resulting dataset for exporting to TagFlip. At this point, we had 300,500 songs and 362 tags left. The top 100,000 songs of this set in terms of number of tags were chosen to be used in TagFlip. With this subset, the number of tags was reduced to 358.