Task 3.2: Add a tag-based search page - 20P

WHAT ARE THE SHORTCOMINGS OF OUR VERY SIMPLE *TAGGING* APPROACH?

It does not support wildcards (but could be realized easily).
Any tag can be created which may include typos, synonymy (multiple tags having the same meaning), polysemy (tags having multiple different meanings) and noise (tags which just don't make any sense related to the media) which might lead to problems in later processing.

WHAT CAN BE SAID ABOUT RELATIONSHIPS BETWEEN TAGS?

Edges represent relationships between tags. From center outwards we find all possible tags from our media library. Each leaf from our graph always represents a file itself which is then connected to all its tag nodes (so called Folksonomie can be any imaginable description of the media related mood, context, genre, style, ...).
The more outside a leaf is located, the less edges it has, which in turn implies less tags, less gravity (in the spring model) and less similarities to other media files.

WE USED TAGS FOR SIMPLE "CONTENT TYPING". HOW CAN CONTENT TYPING IN THE SENSE WE DID IT IMPROVE NAVIGATION AND SEARCH IN AN INFORMATION SYSTEM?

Finding files will be easier, faster, … Searches can be automated based on domain specific needs.

Task 3.3: Metadata extraction 2

DISCUSS THE ADVANTAGES AND LIMITATIONS OF YOUR HEURISTICS.

Advantages:

- Understandability: The logic behind the dominant color algorithm is quite easy to understand and read.
- Dependencies: The function `tagImageByColor` does not require any external libraries.

Limitations:

- Performance: As the algorithm looks at every single pixel, it is rather slow and inefficient when large pictures are to be analyzed.
- Accuracy: The threshold for adding tags for non-maximum colors is set according to the number of pixels times color components for each image. If the image contains large fields of black and/or white, the image is likely to be tagged as red, green and blue.

DISCUSS THE ADVANTAGES AND LIMITATIONS OF OUR SIMPLE DOMINANT COLOR MODEL.

The dominant color model is RGB.

Advantages:

- Popularity: RGB is a very widespread color model, thus the algorithm is guaranteed to work with most images.
- Simplicity: RGB is a very simple and intuitive color model.

Limitations:

- Luminance and Chrominance: RGB is not suitable for extracting color and lightness information separately, what distinguishes it from other color models like YC_bC_r.

Task 3.4: Expose your data in Graph Dracula

WHAT (TYPE OF) LAYOUT ALGORITHM DO WE USE FOR THE INFORMATION VISUALIZATION IN THIS ASSIGNMENT?

The algorithm used is called "Spring"; an algorithm for force-directed graph drawing.

WHAT ARE ADVANTAGES/DISADVANTAGES OF THE USED LIBRARY?

Advantages:

- **Good-quality results**
  At least for graphs of medium size (up to 50–500 vertices), the results obtained have usually very good results based on the following criteria: uniform edge length, uniform vertex distribution and showing symmetry. This last criterion is among the most important ones and is hard to achieve with any other type of algorithm.
- **Flexibility**
  Force-directed algorithms can be easily adapted and extended to fulfill additional aesthetic criteria. This makes them the most versatile class of graph drawing algorithms. Examples of existing extensions include the ones for directed graphs, 3D graph drawing, cluster graph drawing, constrained graph drawing, and dynamic graph drawing.
- **Intuitive**
  Since they are based on physical analogies of common objects, like springs, the behavior of the algorithms is relatively easy to predict and understand. This is not the case with other types of graph-drawing algorithms.
- **Simplicity**
  Typical force-directed algorithms are simple and can be implemented in a few lines of code. Other classes of graph-drawing algorithms, like the ones for orthogonal layouts, are usually much more involved.
- **Interactivity**
  Another advantage of this class of algorithm is the interactive aspect. By drawing the intermediate stages of the graph, the user can follow how the graph evolves, seeing it unfold from a tangled mess into a good-looking configuration. In some interactive graph drawing tools, the user can pull one or more nodes out of their equilibrium state and watch them migrate back into position. This makes them a preferred choice for dynamic and online graph-drawing systems.
- **Strong theoretical foundations**
  While simple ad-hoc force-directed algorithms often appear in the literature and in practice (because they are relatively easy to understand), more reasoned approaches are starting to gain traction. Statisticians have been solving similar problems in multidimensional scaling (MDS) since the 1930s, and physicists also have a long history of working with related n-body problems - so extremely mature approaches exist. As an example, the stress majorization approach to metric MDS can be applied to graph drawing as described above. This has been proven to converge monotonically. Monotonic convergence, the property that the algorithm will at each iteration decrease the stress or cost of the layout, is important because it guarantees that the layout will eventually reach a local minimum and stop. Damping schedules cause the algorithm to stop, but cannot guarantee that a true local minimum is reached.

Disadvantages:

- **High running time**
  The typical force-directed algorithms are in general considered to have a running time equivalent to $O(n^3)$, where n is the number of nodes of the input graph. This is because the number of iterations is estimated to be $O(n)$, and in every iteration, all

pairs of nodes need to be visited and their mutual repulsive forces computed. This is related to the N-body problem in physics. However, since repulsive forces are local in nature the graph can be partitioned such that only neighboring vertices are considered. Common techniques used by algorithms for determining the layout of large graphs include high-dimensional embedding, multi-layer drawing and other methods related to N-body simulation. For example, the Barnes–Hut simulation-based method FADE can improve running time to $n * log(n)$ per iteration. As a rough guide, in a few seconds one can expect to draw at most 1,000 nodes with a standard n2 per iteration technique, and 100,000 with a $n * log(n)$ per iteration technique. Force-directed algorithm, when combined with a multilevel approach, can draw graphs of millions of nodes.

- **Poor local minima**
  It is easy to see that force-directed algorithms produce a graph with minimal energy, in particular one whose total energy is only a local minimum. The local minimum found can be, in many cases, considerably worse than a global minimum, which translates into a low-quality drawing. For many algorithms, especially the ones that allow only down-hill moves of the vertices, the final result can be strongly influenced by the initial layout that in most cases is randomly generated. The problem of poor local minima becomes more important as the number of vertices of the graph increases. A combined application of different algorithms is helpful to solve this problem. For example, using the Kamada–Kawai algorithm to quickly generate a reasonable initial layout and then the Fruchterman–Reingold algorithm to improve the placement of neighboring nodes. Another technique to achieve a global minimum is to use a multilevel approach.

Quelle: https://en.wikipedia.org/wiki/Force-directed_graph_drawing#Advantages