Cluster-Based Information Retrieval in Tag Spaces

Damir Vandic, Jan-Willem van Dam,
Flavius Frasincar, and Frederik Hogenboom

Econometric Institute

ERASMUS UNIVERSITEIT ROTTERDAM

Introduction

- Many Web services enable users to label content on the Web by means of tags (Flickr, Delicious, etc.)
- No restrictions on tags and thus prone to errors and ambiguity
  - typographical errors and syntactic variations, i.e., different tags having the same meaning (e.g., waterfall, waterfal, water-fall, etc.)
  - synonyms, i.e., semantic relatedness (e.g., interior, inside, indoor, etc.)
  - homonyms, i.e., tags that have multiple meanings (e.g., apple, orange, mouse, etc.)
- Solution to these issues: the Semantic Tag Clustering Search framework (STCS), consisting of three parts
  - removing syntactic variations
  - semantic clustering
  - improving search and exploration

Removing syntactic variations

- Input for the algorithm is a graph $(\mathbf{T}, \mathbf{E})$ where the vertices $\mathbf{T}$ are the tags and the edges $\mathbf{E}$ are weights, defined as

$$\begin{aligned} w_{ij} &= z_{ij} \times (1 - lv_{ij}) \\ &+ (1 - z_{ij}) \times \cos(\text{vector}(i), \text{vector}(j)) \end{aligned} \qquad (1)$$

  where

$$z_{ij} = \frac{\max(\text{length}(t_i), \text{length}(t_j))}{\text{length}(t_k)} \in (0, 1] \qquad (2)$$

  and

$$t_i, t_j, t_k \in \mathbf{T},\ \text{length}(t_k) \geq \text{length}(t)\ \forall t \in \mathbf{T},$$

- The algorithm cuts edges that are below a threshold; the remaining components in the graph are the clusters

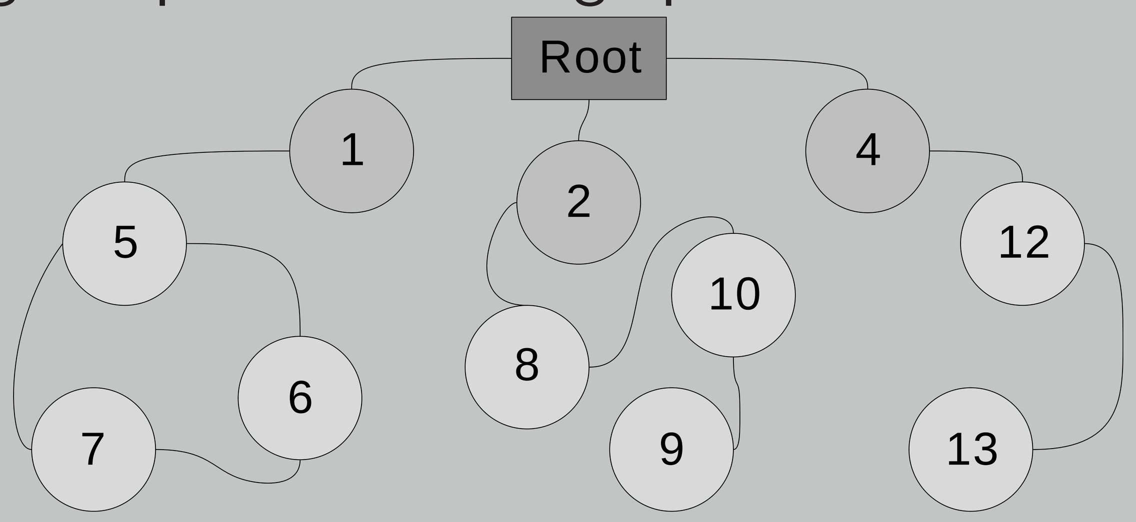

Figure: An example of an input graph for the syntactic variation clustering algorithm

- Dealing with short tags
  - consider the candidate tags 'walk' and 'wall'; normalized Levenshtein similarity is $1 - 1/4 = 3/4$, which is high
  - the cosine similarity of 'walk' and 'wall' is low, so it is a corrective measure for this issue with short tags; shorter tags get a high weight for the cosine similarity
- Heuristic for dealing with numbers in tags: cut an edge if the alphabetic part is the same but the numeric part is not
  - for example, the edge between 'Canon EF 24-105mm f/4 L IS USM' vs. 'Canon EF70-200mm f/4L IS USM' would be cut, because '241054' differs from '702004' and the alphabetic parts are the same

Semantic clustering

- Semantic relatedness between tags is obtained using the cosine similarity based on co-occurrence vectors
- We consider non-hierarchical clusters, adapted version of the method proposed by Specia and Motta (2007)
- The semantic clustering algorithm consists of two steps:
  - create initial clusters
  - merge similar clusters by using the merging heuristics
    - (1) $\mathbf{c} \subseteq \mathbf{C}$, (2) $\mathbf{avgcosine}\,(\mathbf{c}, \mathbf{C}) > \delta$, and (3) $\mathbf{normdiff}\,(\mathbf{c}, \mathbf{C}) < \varepsilon$
    - $\mathbf{avgcosine}\,()$ is the average cosine of all $\mathbf{c} - \mathbf{C}$ and $\mathbf{C}$
    - $\mathbf{normdiff}\,()$ is percentage difference between the clusters w.r.t. the larger cluster, the ε is defined as

$$\varepsilon = \frac{\phi}{\sqrt{|C|}} \qquad (3)$$

Improving search and exploration

- If a tag appears in multiple clusters (i.e., has multiple meanings), the user can choose one cluster
- The semantic clusters are used to sort the pictures:

$$\mathbf{g}(\mathbf{q}_i, \mathbf{p}) = \frac{1}{\mathbf{n} \times |\mathbf{C}_i|} \sum_{c_j \in C_i} \sum_{k=1}^{n} \cos(\mathbf{c}_j, \mathbf{p}_k) \qquad (4)$$

  where $\mathbf{q}_i$ is a query tag, $\mathbf{p}$ is a picture, $\mathbf{p}_k$ is tag $\mathbf{k}$ from picture $\mathbf{p}$, and $\mathbf{C}_i$ is the chosen cluster for query tag $\mathbf{q}_i$
- Example screenshot of implementation:

Figure: This example shows the multiple meanings the tag 'apple' can have

Evaluation

- Syntactic variation algorithm: precision is 0.95 for a test set of 100 tags (all combinations)
  - examples: 'flat-coated retriever' and 'flatcoatedretriever', 'turquoise' and 'turqoise', and 'autumn' and 'automne'
- Semantic clustering: on a test set of 100 clusters, the STCS version has 0.90 precision vs. 0.87 precision of the original method
  - it also finds translations of tags, an example: {'paris', 'frankreich', 'francia'}
  - the K-means algorithm shows 0.84 precision for the cosine similarity on the same test set
- Improving search and exploration: results show that cluster-based search methods are useful

vandic@ese.eur.nl