MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Ruse

Information Retrieval

Yordan Kalmukov

May 2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Describing objects by keywords.
Searching by explicit selection of keywords.

I. Describing objects by unordered set of keywords

Objects are described separately and independently of each other using an ***unordered set of keywords***. In some literary sources, it is also called a *list of keywords*, since the concept of a list is more familiar and close to users. However, from a technical point of view, this is not correct, because the list structure is essentially ordered.

In the beginning, before starting to add documents, the administrator defines a set of keywords. Their number should be as many as necessary, but not too many. When adding a new document to the collection, the user selects only those keywords that most accurately describe it.

The method can be implemented in the following two ways:

1. By selecting the keywords using HTML checkboxes (Figure 1). In this case, the keywords have a "binary" behavior - each of them is either selected or not; either present in the object description or not.
2. By selecting the keywords using HTML drop-down menus (select boxes). For each keyword, there is a drop-down menu (Figure 2), which allows the user to indicate not only whether it is applicable to the description of the corresponding resource, but also to state exactly how applicable it is. This introduces “degree of applicability / significance” of the given keyword to the description of the specific object.

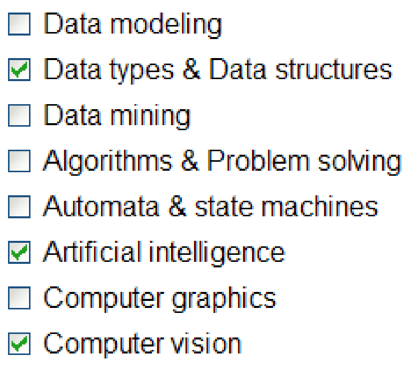
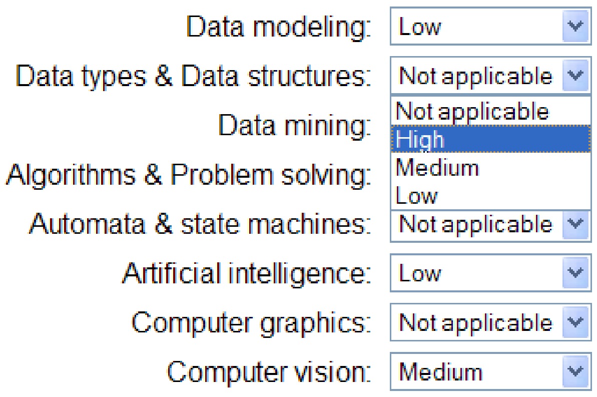

☐ Data modeling
☑ Data types & Data structures
☐ Data mining
☐ Algorithms & Problem solving
☐ Automata & state machines
☑ Artificial intelligence
☐ Computer graphics
☑ Computer vision

Figure 1. Describing an object by selecting keywords from unordered list (by HTML checkboxes). The keywords have a binary-like behavior – selected or not selected.



Data modeling: Low

Data types & Data structures: Not applicable

Data mining:

Algorithms & Problem solving:

Automata & state machines: Not applicable

Artificial intelligence: Low

Computer graphics: Not applicable

Computer vision: Medium

Figure 2. Describing an object by weighted keywords, selected by HTML select boxes. If a keyword is selected the user can state how much it applies to the document (object). "Not applicable" means not selected.

If the keywords are selected from checkboxes, there are several easy options to calculate the similarity coefficients between any two documents (or between the query and each of the documents):

1. **Simple match**

$$Sim(d_i, d_j) = |KW_i \cap KW_j| \tag{1}$$

where:

$Sim(d_i, d_j)$ – similarity between documents d_i and d_j.

KW_i – set of keywords, describing the *i*-th document.

KW_j – set of keywords, describing the *j*-th document.

Equation (1) return the number on common keywords between the two sets.
The symbol ∩ means intersection, and the vertical lines returns the number of elements.

The problem with simple matching is that the resulting value is not normalized to a certain interval, say [0, 1]. If you have two objects described with 2 exactly matching keywords, and another two objects described with 3 exactly matching keywords, it turns out that the second pair of objects is actually more similar than the first pair. This, of course, is not true, because in both pairs, the similarity is 100%.

2. **Jaccard's index**

$$Sim_{Jaccard}(d_i, d_j) = \frac{|KW_i \cap KW_j|}{|KW_i \cup KW_j|} \tag{2}$$

Here, all the notations are already described.

The Jaccard's index calculates the degree of similarity as the ratio of the number of common keywords to the number of all unique (for both sets) keywords.

Its value is normalized to the interval [0, 1], which is an advantage over simple matching.

3. Dice's coefficient

$$Sim_{Dice}(d_i, d_j) = \frac{2 \times |KW_i \cap KW_j|}{|KW_i| + |KW_j|} \tag{3}$$

According to the Dice coefficient, similarity is calculated as *2 times the number of common keywords divided by the number of all (for both sets) keywords*. Duplicates are not removed in the denominator (as in Jaccard)!

If there are **5** keywords in the first set, and **3** in the second, and **2** of them match between the sets, then the denominator for Dice will be 8 (5+3), and for Jaccard's index - 6 (5+1). With Jaccard's index it will be 5+1 because 2 of the keywords in KW_j are already in KW_i.

I.e. in the described case, the similarity according to Jaccard's index will be 2/6 = 0.33, and according to Dice's coefficient- 4/8 = 0.5.

Obviously, the Dice coefficient is also normalized in the interval [0, 1].

Saif Mohammad and Graeme Hirst proved that the relationship between two similarity factors calculated by Jaccard's formula is preserved when they are recalculated by Dice's formula as well. Yes, there is a difference in absolute values, as we saw in the example above, but the relationships between the individual coefficients are preserved. Therefore, if you are not interested in the absolute values, but only in the relationships between the coefficients, then it does not matter whether you use the Jaccard or Dice similarity measure. When searching and ranking the results, we are generally not particularly interested in the absolute values of the similarities, but only the relationships between them. Results are sorted based on relationships. Biggest similarities go higher in the result list. No matter what the exact value is.

In the literature, the Dice coefficient could be also found as the Sørensen index. The two scientists proposed it independently of each other, but Dice published it first.

If keywords are weighted (as in Figure 2), it is mandatory that the similarity measure not only take into account the number of matching/common keywords, but also their respective levels. How can this happen? Well - when 2 keywords (one of a set) match but their relevance levels are not maximum, you won't count the match as 1 full match. You will count it as 0.x matches by decreasing the 1 inversely proportional to the applicability level, i.e. the greater the applicability of the word, the less you will reduce.

$$|KW_i \cap KW_j| => \sum_{k_m \in KW_i, k_n \in KW_j} (1 - (1 - w_m) - (1 - w_n))$$

II. Describing documents/objects by taxonomy of keywords

Selecting keywords from *a predefined, unordered set* individually and independently describes all documents. But the unordered nature of the set requires that *its size be limited to a reasonable (not large) number of semantically non-overlapping elements* - say 20 to 30. Otherwise, the user interface will be very inconvenient to work with. If the set of keywords *contains hundreds of items and they are not structured* in any way, it will take a *very long time for the user to read them* and select the ones that best describe the relevant resource. This could greatly demotivate him/her and he/she might give up or make superficial choices. On the other hand, *the small number of keywords leads to a lack of specificity (details)* in them or an inability to fully cover the thematic areas of the documents.

The stated problem can be largely solved, if the selection of keywords is done not from an unordered set, but from a ***taxonomy***. The advantage of this method is a direct consequence of the *hierarchical structure* of the taxonomy. It provides additional and very important information - the semantic relationships between individual keywords, which allows:

1. Similarity measures to consider ***not only the number of exactly matching keywords***, but also the ***semantic similarity between non-matching ones***.
2. To calculate a ***non-zero similarity*** between two documents, even when ***they do not share any common keyword***.
3. The taxonomy may include many more keywords - hundreds, even thousands. Their larger number provides a ***more detailed and accurate description*** of the objects, without causing inconvenience when working with the user interface. The elements are grouped into generalized branches in the tree, and the user "opens" only those branches that interest him.

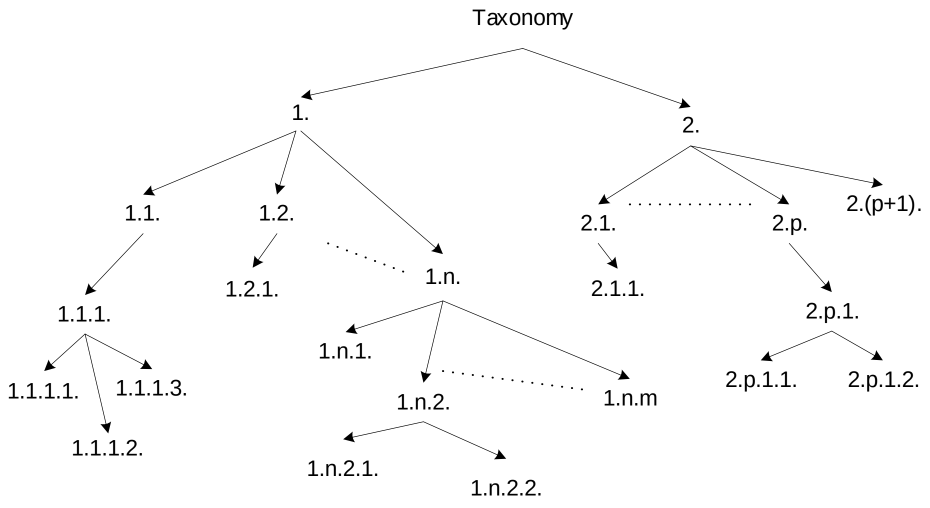


Figure 3. General structure of the taxonomy

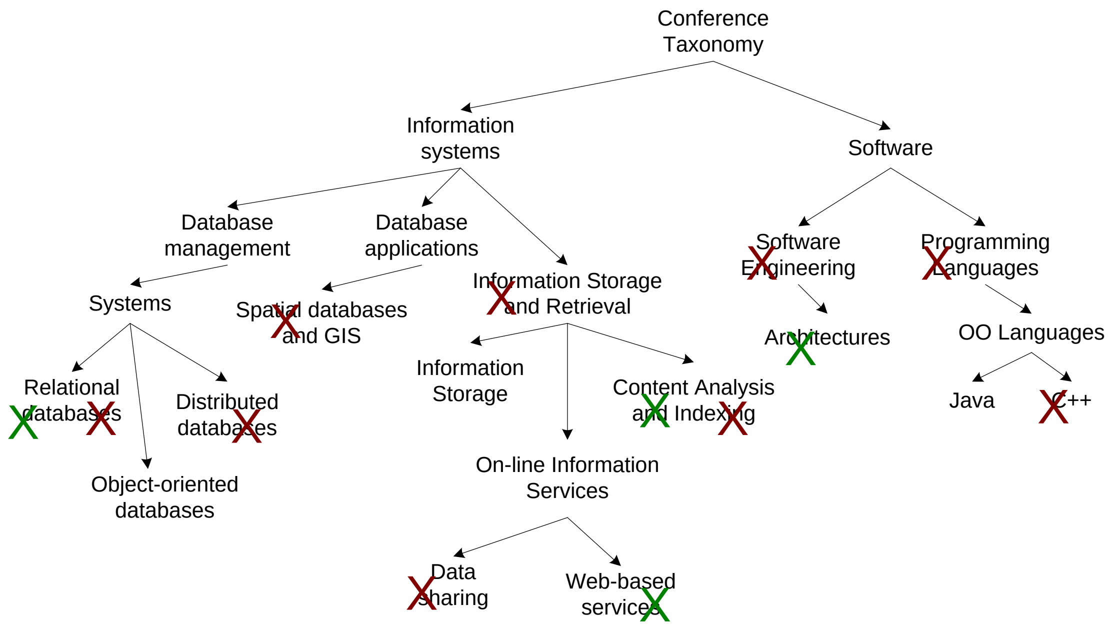


Figure 4. Example taxonomy with the keywords describing the first document (colored in green) and the second document (colored in dark red).

According to the figure, the keywords that describe the two documents are:

KW_i = {Relational databases, Content analysis, Web-based services, Architectures}

KW_j = {Relational databases, Distributed databases, Spatial DB & GIS, Information storage and retrieval, Content analysis, Data sharing, Software Engineering, Programming languages, C++}

Although they are stored in unordered sets, the semantic relationships between them are preserved and could be extracted from the taxonomy at any time.

Documents are typically described not by one, but by multiple keywords selected from the predefined taxonomy. Therefore, in order to accurately calculate the degree of similarity between them, it is necessary to use a similarity measure that calculates the semantic similarity between two sets of nodes (concepts) in a common taxonomy. I proposed the one below (4). It is based on and derived from Dice's coefficient. However, instead of the number of exactly matching keywords, it also counts the semantic similarity between the non-matching ones. How is this possible? Because the keywords are not independent, but hierarchically related within the taxonomy. The semantic similarity between any two keywords could be calculated based on the length of the path between them or in other ways.

$$Sim(d_i, d_j) = \frac{\sum_{k_m \in KW_i} \max_{k_n \in KW_j} (Sim(k_m, k_n)) + \sum_{k_n \in KW_j} \max_{k_m \in KW_i} (Sim(k_n, k_m))}{|KW_i| + |KW_j|} \quad (4)$$

where:

k_m – m-th keyword, describing the i-th document.

k_n – n-th keyword, describing the j-th document.

KW_i – set of keywords, describing the *i*-th document.

KW_j – set of keywords, describing the *j*-th document.

$Sim(k_m, k_n)$ – semantic similarity between the *m*-th keyword, describing the *i*-th document and the *n*-th keyword, describing the *j*-th document.

$\max_{k_n \in KW_j}(Sim(k_m, k_n))$ – semantic similarity between the *m*-th keyword, describing the *i*-th document and its semantically closest keyword, describing the *j*-th document.

If the taxonomy is converted to an unordered set, by ignoring the semantic relations between the individual elements, then formula (4) will always give exactly the same result as the Dice's coefficient (3). But unlike it, (4) can also be used for sets whose elements are semantically related.

To calculate $Sim(d_i, d_j)$ one must first compute all similarities $Sim(k_m, k_n)$ between all keywords describing one document and all keywords describing the other document. This can be done in one of two ways:

- Based on the structural characteristics of the taxonomy - distance, depth, density, etc.
- Based on the information content of the nodes.

One of the widely used measures to determine the semantic similarity between two nodes in a taxonomy is that formulated by Zhibiao Wu and Martha Palmer (5).

$$Sim_{Wu\ \&\ Palmer}(k_m, k_n) = \frac{2 \times N_0}{2 \times N_0 + N_1 + N_2} \tag{5}$$

where:

N_0 - distance (in number of edges) between the root and the closest common ancestor C_0 of the two nodes/concepts (C_m) and (C_n) (fig. 5).

N_1 - distance from C_0 to one of the concepts, for example C_m. C_m represents the *m*-th keyword from the *i*-th set.

N_2 - distance from C_0 to the other concept – C_n. C_n represents the *n*-th keyword from the *j*-th set.

Since the similarity measure of Wu and Palmer is symmetrical, it does not matter whether C_m belongs to the *i*-th set, and C_n to the *j*-th or the opposite. A closer look at (5) shows that it actually represents a Dice coefficient applied to the sets of edges, building the paths from the root to the two nodes/concepts between which semantic similarity is sought.

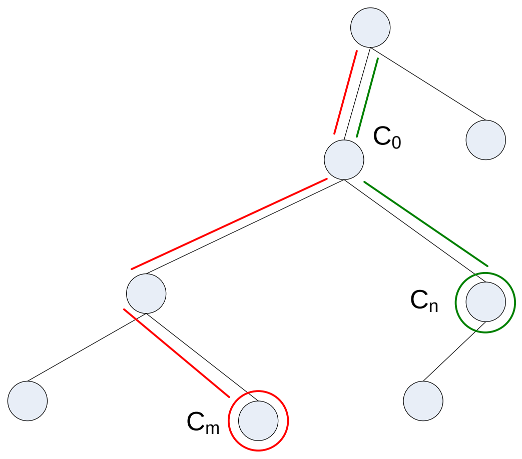


Figure 5. Visual representation of the similarity measure of Wu and Palmer for calculating similarity between two concepts in a taxonomy

Dekang Lin proposes a similarity measure based on concepts' information content. In this case, the taxonomy is augmented with a function *p:C* -> [0,1], so that for any node (concept) $c \in C$, *p(c)* represents the probability of encountering the node *c* or any of its descendants in the taxonomy. I.e. if c_1 is in "IS-A" relationship with c_2, then $p(c_1) <= p(c_2)$. It follows that the probability of the root (if any) is 1 because every node is its successor. Since lower probability means higher information content, nodes deeper in the hierarchy are more informative than those located shallower. Lin's similarity measure (6) is similar to Wu and Palmer's, but instead of distances, it considers the information content of nodes.

$$Sim_{Lin}(k_m, k_n) = \frac{2 \times \log P(C_0)}{\log P(C_m) + \log P(C_n)} \tag{6}$$

where

$P(C_0)$ - probability of encountering the closest common ancestor C_0 or any of its descendants in the taxonomy.

$P(C_m)$ - probability of encountering the concept (or its successor) representing the *m*-th keyword from one set.

$P(C_n)$ - probability of encountering the concept (or its successor) representing the *n*-th keyword from the other set.