Categorical representation of seaman's knots

Miguel Brito Franco

Department of Computer Science and Engineering, Instituto Superior Técnico

Abstract

Knot representation has been documented by seamen for some centuries and, despite the topic's significance, a categorical representation of knots through an expert system is not often addressed. In a universe of over 3800 possible knots, knowing which one to use in a specific situation is not always trivial. Using tree clustering to create a taxonomy, the system enables queries to be performed over the dendrogram, taking advantage of two levels of abstraction, and admitting a possibility of uncertainty in the search. With the seaman's knot categorical representation system, it is expected that a non-expert user can easily know which knot to use on a boat or in a specific situation, like when patching a sail or when rescuing someone thrown overboard.

Keywords: Categorization; Clustering; Uncertainty; Taxonomy; Dendrogram; Expert System; Knot.

1. Introduction

Knots are used since immemorial times and play an important role in human history (Cotterell & Storm 2003; Silverstein 2012). However, it is a difficult task for a non-expert person to identify a knot. The same knot can have different names, two or more knots can have the same functional characteristics and they can be used for the same utility purposes. This subject is not often addressed by computer science. The knowledge representation and categorization problem is not trivial to address since the knowledge is sensitive to its domain. One of the possible solutions to address knowledge representation is the usage of a category structure (Wichert 2000). In the domain of seaman's knots, this approach makes a particularly sense, since knots can be described through internal characteristics, like 'safe', 'easy to untie', or 'ornamental', and usage characteristics, like 'used in a sail', or 'around a tool'. An inexistence of a system that models the expertise of a seaman on knots and lets a user address those knots motivated us to make this solution.

We focused in this problem and we wanted to verify if the hierarchical categorization is a reliable approach to represent the seaman's knowledge in the knots domain and if this knowledge is easily addressable and retrievable. The purpose of this system is to categorize seaman's knots by their features in a taxonomy structure and allow their retrieval based on the belief of presence or absence of features. We intend the user to be able to query a taxonomy of knots, which was created through hierarchical clustering, by taking advantage of two levels of features' abstraction, and admitting a possibility of uncertainty in the search.

We also introduce a new form of dealing with uncertainty, based in quantum computation. The method will allow the retrieval of knots based on the belief of having features, aggregated with a probability.

2. Knowledge Representation

2.1. Concepts

Think process is the way on how we represent the world and manipulate those representations (Osherson et al. 1998). To make a system where we can manipulate representations, we have to make a conceptual schema. This conceptual schema unifies database, applications, and user interface (Sowa 1999), Figure 1.

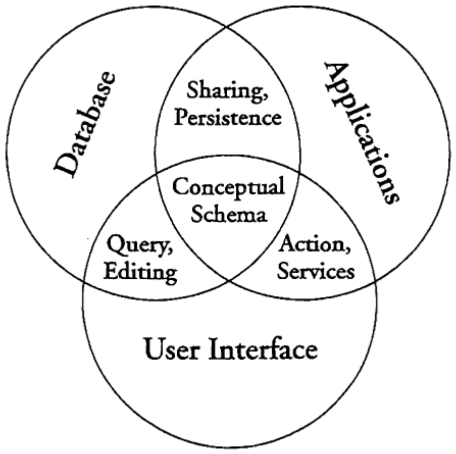


Figure 1 - Conceptual Schema (Sowa 1999)

This conceptual schema can be applied to an expert system, which is a system that filters significant criteria (Otman 1996) in order to retrieve knowledge. A frequent way to represent knowledge is through a taxonomy (Lucas & van der Gaag 1991; Wichert 2000), which is a hierarchical representation of related concepts.

Similarity between objects is assumed to increase with the number of shared features (Osherson et al. 1998) and a way to calculate the similarity is through the Tversky similarity function (Tversky 1977; Wichert 2000) where B is the feature vector and Ca the category:

$$Simc(Ca, B) = \frac{|Ca \cap B|}{|Ca|} - \frac{|Ca - (Ca \cap B)|}{|Ca|}$$

When category (Ca) is constant, the function equals the Jaccard similarity function (Jaccard 1901) between two vectors:

$$Jaccard(v_1, v_2) = \frac{|v_1 \cap v_2|}{|v_1 \cup v_2|}$$

This similarity is used to compare two vectors, like binary vectors. These are arrays of dimension n where the values vary between 0 and 1. The value 1 denotes a presence, and the value 0 denotes an absence.

If one uses a taxonomy, one will get a hierarchical structure. To this hierarchical structure is possible to apply a decision tree where, in each node, a decision is made to choose between branches of the tree (Russel & Norvig 2010; Quinlan 1986; Smith 2010).

2.2 Clustering

Clustering (Gorunescu 2011) is the process of grouping objects that aggregate a same set of characteristics. Those aggregations are called clusters. Objects are similar inside a cluster and dissimilar with objects from other clusters.

Distance between clusters can be measured with minimum distance between two clusters, also called nearest-neighbour.

One of the technics for form clusters is through an agglomerative hierarchical clustering (Everitt 2011; Liu et al. 2000).

This algorithm starts by letting all the unities of knowledge, in our case the knots, to be in individual clusters. Then, after measuring the most similar clusters, it merges them.

An inherit problem of clustering is the Curse of Dimensionality (Beyer et al. 1999). This problem states that, as dimensionality grows, the distance between the various data records gets progressively similar, i.e. the distance difference from the nearest neighbor to the farthest neighbor gets drastically the same. Such problem can put the clustering technics to waste when a big dimensionality is reached. We will later see that this problem is eased by pre-agglomerating the knots into sub categories.

3. Solution Proposal

The purpose of this system is to categorize seaman's knots by their features in a taxonomy structure and allow their retrieval based on the belief of presence or absence of features.

The various steps to accomplish this objective are described below. They include the formation of the dataset and how the data is structured, the hierarchical clustering algorithm, the process of retrieval, and the role of uncertainty in dealing with that retrieval process.

In order to achieve a categorical representation that can be later manipulated to retrieve results, one can divide the system in two distinct moments: the actual construction of the data and its categorization, and the retrieval process that allows the user to consult the data and achieve positive results.

3.1 Overview

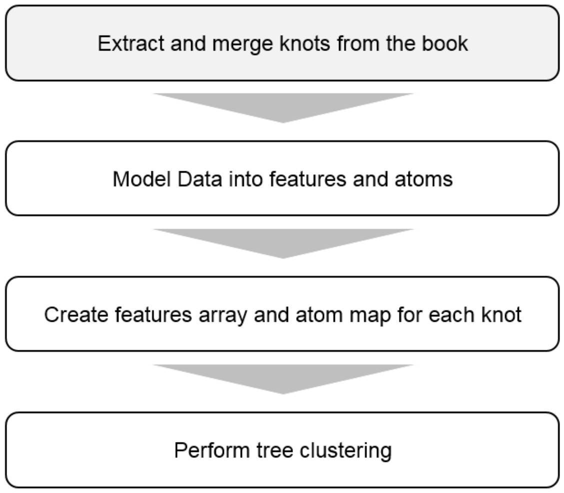

Extract and merge knots from the book
Model Data into features and atoms
Create features array and atom map for each knot
Perform tree clustering

Figure 2 - View on the creation of the knot's data

Figure 2 represents the data organization. First, a dataset was created resorting to a book of knots and though the extraction of their features. After merging the unique knots into single entries, the knot's data could be divided into two levels of resolution, a more general of features and a more specific of atomic features. In order to have a representation that is easily manageable, the knots are represented by features in a binary array and the atomic features in a map structure where a feature is the key and a binary array of atomic features is the value. Finally, the data is ready to be categorized in a form of a tree cluster, also called as a dendrogram. The final result is a taxonomy of knots that share equal features and is able to be searched.

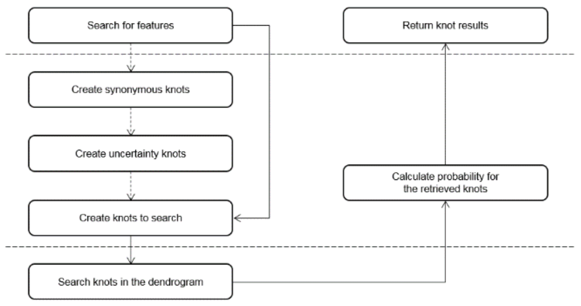


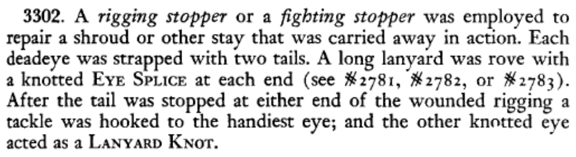
Figure 3 - View on the layers of the retrieval process

This search is performed by a user that wants to know a knot based on its features. As one can see in Figure 3, there are three layers of abstraction to the implemented code.

The user is only able to search for features and expect the return of a knot that has those features. This is called the presentation layer.

In the logic layer, the query made by the user takes a form that can be interpreted by the system, in order to retrieve a set of knots. In this layer the user's input is chunked into features, atomic features, and uncertain values. For each one of these, there is a confirmation on either the term is present in the characterization of any knot. If the term is not present, a synonymous term is created. Then the uncertainty of possible values is considered. This uncertainty allows the user to search for terms that might or might not be present in the knot. Finally, the user's query is in the same formal form as the knots in the dendrogram.

3.2 Domain



3302. A *rigging stopper* or a *fighting stopper* was employed to repair a shroud or other stay that was carried away in action. Each deadeye was strapped with two tails. A long lanyard was rove with a knotted EYE SPLICE at each end (see #2781, #2782, or #2783). After the tail was stopped at either end of the wounded rigging a tackle was hooked to the handiest eye; and the other knotted eye acted as a LANYARD KNOT.

Figure 4 - Entry from the Ashley Book of Knots (Ashley 1944)

In principle, the process of identifying seaman's knots by their names and description can be quite straightforward as we can see through Figure 4. Anyhow, there are two inherent difficulties on creating this new dataset. In first place, the book didn't present itself in a form of editable text, raising the difficulty of automatically extract hundreds of entries from the book. To accomplish this challenge, a group of people was asked to introduce knots into a table. They had to introduce both the entry numbers and the names corresponding to the knots. This knots were later merged by their name.

Another difficulty is a problem that is inherited from Natural Language, linguistic variability, *i.e.* same features can be expressed by different words or sentences.

In order to ease this problem, features are structured in a relation that is later abstracted to create a taxonomy.

Features and Atomic Features

Motivated by the usage of semantic networks (Steel 2012; Novak 2010) and in how the human brain aggregates similar concepts (Huth et al. 2016), we decided to divide the similar concepts into groups. These groups are then divided into two levels of abstraction, Features and Atomic Features as it is possible to see in Figure 5.

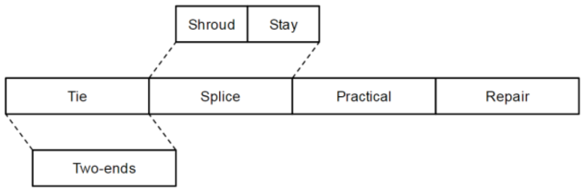


Figure 5 - Array of features and aggregated atom features

By its own an atom feature does not add any valuable information to the knot, only when associated with a feature. Anyhow, setting these two levels of abstraction allow the comparison for similarity between knots to be easier to calculate. Features and Atoms are later used to compare knots and create a hierarchical categorization of knots.

Binary Arrays

[0 1 0 0 1]   { 1 = [1 0 0], 4 = [0 1 1] }

Figure 6 - Features and Atoms in binary representation

In order to reduce complexity in comparing features and atoms, binary arrays were chosen to be another level of abstraction of the conceptual schema. The chosen implementation for this structure is the BitArray (Zhou et al. 2003). Every Knot has a Bit Array of features and a Map of atom features, where the index of the Map is the same index as the feature (Figure 6).

Inverted Index

In order to record each index for every one of the features, atom features, and knot names, we use an inverted indexes. Features an names are stored in LinkedList collections, form the Java™ Standard API. Atoms are stored in Google's® Guava[1] Common Collection Multimap. This allows multiple values to be stored for the same key in a map-like structure.

Tree Clustering

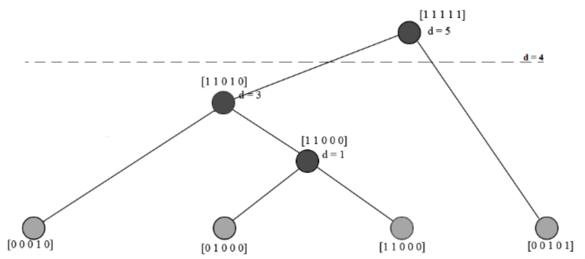


Figure 7 - Dendrogram of four knots

To perform a knot categorization it was used a hierarchical tree clustering. The implemented algorithm was adapted from an existing one (Johnson 1967; Fournier-Viger 2014). Even though the original code was severely rewritten to fit the purposes of the domain, methods for time and memory measurement were kept.

The cluster uses single-linkage to calculate the distance between the clusters because this measurement minimizes the difference between the features arrays. This is important for the system because we want to have the most similar knots together. Let us consider the following example.

The jug sling is a knot used to carry small loads like a bottle, a jar, a jug, or a bag. This knot can be depicted by the feature "carry" and the atoms "carry.small-load", "carry.bottle", "carry.jar", "carry.jug", and "carry.bag".

The magnus hitch is a knot used to lift large loads, like stones. This knot can be depicted by the feature "carry" and the atoms "carry.large-load" and "carry.stone".

If we measure the distance of the two knots using only the features, this distance is zero because they only have the only same feature "carry". If, on the other hand, we use both the features and the atomic features to measure the distance between the two knots, the conclusion is that distance is 7, one by each of the different atoms.

Both knots serve the same purpose, to carry something, and on a semantic level this is the most important to describe the knot. Considering features and atoms in the measurement of the distance would cause unbalanced results and disfavor knots that are more completely described. Therefore this approach was not considered.

Furthermore, the dimensionality problem that is aggregated with the clustering algorithms is eased with this approach. The dimension of the knots is immediately reduce by not considering the atom features and, by aggregating similarities a priori, is possible to reduce the dimensionality without losing any information.

An example of a tree cluster can be viewed in Figure 7.

Distance Functions

The feature and atom arrays that characterize each knot have always a fixed length, namely 147 for the feature array and 30 for the atom array. This numbers correspond to the total number of

[1] https://github.com/google/guava

features and to the maximum number of atoms for one feature, respectively.

Hamming distance function (Hamming 1950) is a function that allows to measure the difference between two arrays of the same length. For every position ([i, n-1]) of an array (v) the Hamming distance is the sum of the module of the difference between each position of the array.

$$Hamming(v_1, v_2) = \sum_{i=0}^{n-1} |v_{1_i} - v_{2_i}|$$

In particular, the result of the Hamming distance for bit arrays varies between '0' – if the bits in the same index of the two arrays are the same – and '1' – if the bits in the same index of the two arrays are the different. This is the same as performing a XOR between the two arrays.

$$Hamming\ (v_1, v_2) = XOR(v_1, v_2)$$

The Euclidean distance is default way to measure the distance in a Euclidean space. This value is also the shortest distance between two point in a Euclidean space and is calculated through the following formula,

$$Euclidean\ (v_1, v_2) = \sqrt{\sum_{i=0}^{n-1} \left(v_{1_i} - v_{2_i}\right)^2}$$

Where 'n' is the dimension of the array, 'i' is the index of the array and 'v1' and 'v2' are the arrays. Knowing that the distance in each index is either '0' or '1', and likewise the Hamming distance, the Euclidean distance can be calculated has the root of the XOR of the two arrays. Therefore,

$$Euclidean(v_1, v_2) = \sqrt{Hamming(v_1, v_2)}$$

It is possible to assume a '0' distance between knots. In the context of the work, different knots can serve the exact same purpose and have the exact same features and atoms.

It is also possible to define a threshold for the maximum distance. After this distance threshold, two knots don't belong to the same cluster.

3.2 Retrieval

After executing the categorization process we are able to perform queries over the dendrogram. A user is able to search for knots by providing the features. When a query is performed the given features are present and all the others are assumed to be absent.

Single Set Rule

Independently of the probability of success, if there is only one option in the set, the knot on that set is the right answer, with 100% of certainty.

Query for Synonymous Words

One of the weaknesses of feature indexing is that synonymous words are not considered. This leads to a limitation in the lexicon that a user can query in the knot retrieval, e.g. for a knot that is characterized by a feature "useful" we could say that the knot is also "handy" or "helpful".

In order to take advantage of the taxonomy retrieval and try to enhance the lexicon the user can apply, it is possible to resort to a database of synonymous. For this purpose, we are employing an online tool, thesaurus[2].

This alternative does not change the existing representation of the knots in the cluster. This happens because:

1 - The usage of inverted indexes allows an easy existence verification for the possible knot's name, features, and atom features.

2 - The query for similar words mimics the short term memory by allowing a verification for possible synonymous. The synonymous words are not added to the inverted index because there could be ambiguity, e.g. the feature "fast" refers to firmly fix the rope to an object but it could be wrongly interpreted as "quick" to tie.

3 - The last and perhaps strongest reason not to use a deductive system is the clustering algorithm. For every time a new rule was added to the long term memory, the dendrogram would have to be recalculated and this would derail any retrieval efficiency.

Decision Tree

The implemented hierarchical clustering algorithm is unsupervised, where all leafs correspond to knots and any tree node indicates the distance between their children. Plus, a node has a Bit Array with all the features of its children. Liu et al. propose a supervised clustering algorithm that divide sparse or dense regions (Liu et al. 2000). In the context of the seaman's knots classification we can apply decisions on each

[2] Thesaurus.com service provided by words.bighugelabs.com

node, even after the unsupervised clustering, and get accurate results in high dimensional datasets.

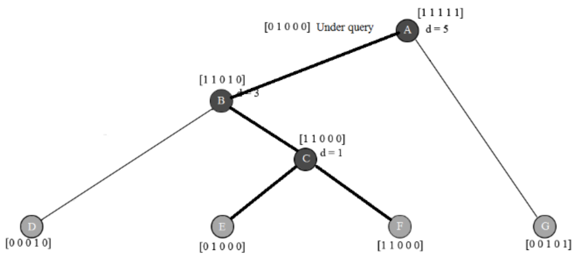


Figure 8 - Search for the knot [01000] in the dendroram

This result is obtained by checking, in each node, if a feature is present. Figure 8 is a visual example on the decision to check the children of each Node.

Uncertainty

Decisions made in every Node in Figure 8 correspond to a search for a knot represented by the features array [0 1 0 0 0]. There is only one present in this array, where all the other four are absent. Now, let us assume the user wants to search for a knot that is 'secure' and it maybe 'temporary'. This search would traduce in a knot that is represented by the array [? 1 0 0 0], where '?' represents a feature that may be present or absent.

Inspired by the uncertainty principles of quantum computing, where instead of bits the positions are represent by quantum bits, or qubits, a unit that may acquire value of 0, 1, or another value between 0 and 1 (Wichert 2014) with a certain probability. Anyhow, when observed the state is either 0 or 1.

The array that is being searched would take both the form of [0 1 0 0 0] and [1 1 0 0 0], because the feature 'temporary' can be both present and absent in a knot represented in the dendrogram. A priori one does not know what the probability to find the right knot is. One can consult table 1.

Our approach states that is possible to query the dendrogram with uncertainty on how a knot is represented and still get all the possible results. This is possible because all the domain is known for the system even though it is not known for the user. In other terms, the user can search for all the possible knots, without making an assumption on what is the probability of the outcome, this probability of the outcome is calculated and returned by the system in the end of the query.

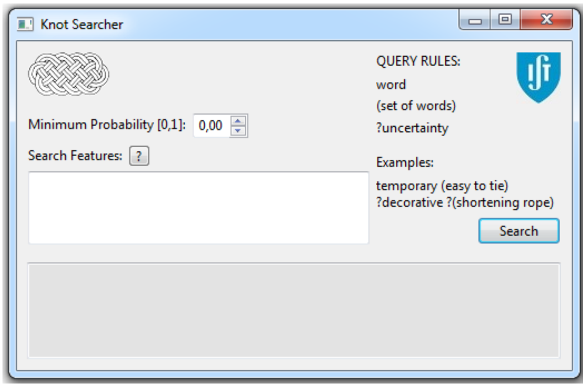
In simple terms, the search is made in parallel, where the query states all the combinations of possible features, and the results are all shown in the end with a certain probability.

	D [0 0 0 1 0]	E [0 1 0 0 0]	F [1 1 0 0 0]	G [0 0 1 0 1]
[0 1 0 0 0]	0	1	0.5	0
[1 1 0 0 0]	0	0.5	1	0

Table 1 - Possible outcomes for [?1000]

Interactive Interface

The user is able to use an interactive interface to search for the knots, instead of the console. It is possible for the user to choose the minimum similarity between the features the user is searching and the knot in the dendrogram.


Knot Searcher

Minimum Probability [0,1]: 0,00

Search Features: ?

QUERY RULES:
word
(set of words)
?uncertainty

Examples:
temporary (easy to tie)
?decorative ?(shortening rope)

Figure 9- User's interactive interface

The main focus of this work is to take advantage of a taxonomy of knots, in order to retrieve them based on their features and atomic features. This section presents the main structure of the solution and details on how its modules permit the achievement of the initial statement. This section is also divided in two moments, the domain construction and the information retrieval.

The domain was built resorting to an existing book, where all the knots had to be introduced by hand in a dataset in order to be later manipulated. The entries in this corpus have information on the entries of the book that describe the knot, on the names of the knot, and on its characteristics. This dataset can be made available for any future work, or for any other work that can also rely on this domain.

To recognize the dataset as valid for all the solution, it was made a conceptual schema with two levels of abstraction, one for features and another for subtypes of this features. These are called atomic features. This conceptual schema is produced with the assumption of representing the human brain categorization of concepts, where concepts of the same origin are grouped together in order to be retrieved more easily.

By representing the features in binary vectors, is supposed a more efficient comparison and retrieval of knots within the taxonomy. Since knots are represented for binary arrays of features, they can be easily compared using logical operators.

It is possible for the user to retrieve the knots by searching for features that are not present in the description of the knot. This approach benefits a wider possibility for the user to find the intended knot. It also introduces error margin that is aggregated with the words' meaning ambiguity.

One of the core aspects of this work is the search for the uncertainty of features. This approach is based on the quantum computing and, by searching the various possibilities in parallel, allows the retrieval of all knot the possibilities with an aggregated probability of certainty.

The user is also allowed to opt between the console interface and a GUI.

In section 4, we perform the evaluation on the solution, stressing its assumptions and assessing the performance of both domain construction and retrieval.

4. Experiments and Results

Domain

One can evaluate that the data is very sparse, where over 90% of the entries of the features arrays is a 0 – in average, for the whole set of knots 90.29% of the features is not present. In average, a knot is defined between 11 and 15 features. Furthermore, as one can consult in Figure 10, 12.4% of the knots have one feature, 27.2% of the knots have 18 features, 35.2% have between 2 and 17 features, and 25.2% of the knots have 19 or more features.

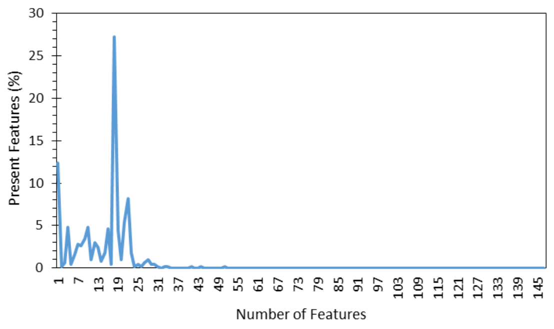


Figure 10 - Distribution of the number of features per knot

Tree Clustering

In order to evaluate if the clustering algorithm is being well performed (Halkidi et al. 2001; Berry & Gordon S. Linoff 2011), we need to validate two aspects, compactness and separation.

Separation measures the distance between clusters and should be as wide as possible. This is achieved and is easily verified when applying a threshold. For every time the distance threshold is met when merging two knots, the two clusters do not merge.

Compactness measures the distance of the members in a cluster and grants that they are as close as possible. To measure this aspect, we use the ward's method, bearing in mind that the objective is to calculate the variance, minimizing it.

Ward's Method

The Ward's (Ward 1963) method is a criterion used to compare clusters in an agglomerative hierarchical structure. It uses the same objective function as k-means, where the total variance around centroids is summed. In each iteration, before merging the two clusters, there is a centroid for each cluster that are being merged. Each of those centroids has a certain deviation to the cluster's points, and the total deviation is the sum of deviations of both clusters. The total distance (TD) of the centroids (C1 and C2) equals the sum of the square distance (D) between the centroid of the merged clusters and the minimum distance of the two centroids (x):

$$TD_{C_1 \cup C_2} = \sum_{x \in C_1 \cup C_2} D(x, \mu_{C_1 \cup C_2})^2$$

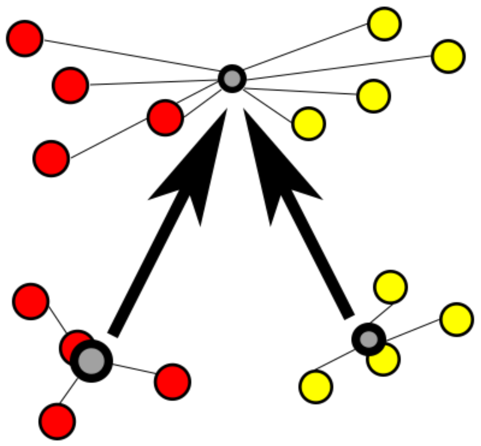


Figure 11 - Centroid shift when merging

Figure 11 is a visualization on how the Ward's method measures the variance of the centroids.

In practice, the sum of these variances represents the sum of the square of the distance error between the centroid and the points in the cluster. For that is also called the Sum of Squared Errors (SSE). Another property in the Ward's method is that it greatly depends on the original clusters that are being hierarchically merged. Even when using the same distance function, number of data points, and threshold, for different datasets the results can be very different, even in an order of magnitude.

The lower the total distance, the more compact is the tree cluster. As one can see in the following Figure 12, the Sum of Squared Errors grow exponentially with the number of knots in a cluster. For a cluster of 50 knots we have, in average, a SSE of 7,962.86. As for the set of 1271 knots we have a SSE of 378,303.50.

By looking at figure 13 we can understand the implementation of the distance functions have a great importance in the overall complexity. Also, we are able to observe that even though the growth of the Euclidean distance appears linear, it really is also exponential.

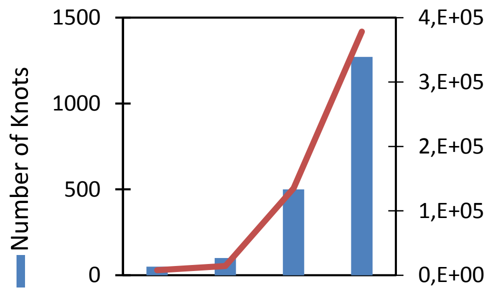


Figure 12 - Growth of SSE with the number of knots

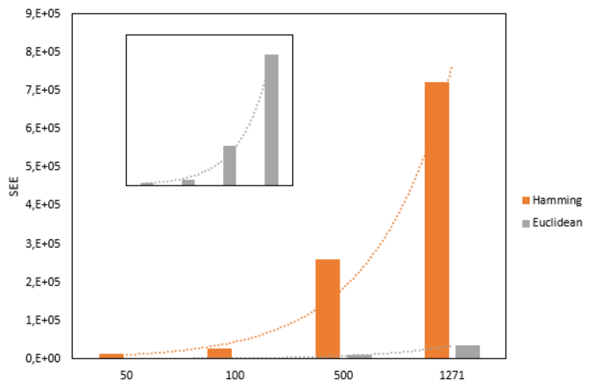


Figure 13 - Hamming and Euclidean distance comparsion of SSE over number of knots

Uncertainty and synonymous searches are found to be well performed. Uncertainty has an exponential growth with the number of uncertain values, where the growing is 2n.

The method to presented by Stillings (Stillings et al. 1987) to evaluate the robustness a knowledge representation system is completely met. This happens because the representation is perspicuous, efficient, uses two levels of abstraction for the features, representational primitives are well defined, knowledge is modifiable to a certain extent, acquisition, retrieval, reasoning are possible, there is a division of labor between the knowledge base and the interpretative program, and representation itself be modified or extended when not in runtime.

In the tree cluster we were able to compare the efficiency of the implementation of the two distance functions. The Euclidean distance, which uses logical operators in the implementation, is always about 22 times quicker than the Hamming distance. Separation of knots is well performed and verified with the increase in the number of clusters with the decrease of the distance threshold. The dendrogram is not very compact because there is a great variance in the features. If one would normalize the distance to be between 0 and 1, the value of the SSE would be smaller, but relatively speaking the compactness would be the same.

The query allows the usage of synonymous words to retrieve a non-declared feature. Uncertainty is possible to reproduce by doubling the number of knots to search for each new uncertain feature that is introduced.

5. Conclusions

We focused in this problem and we wanted to verify if the hierarchical categorization is a reliable approach to represent the seaman's knowledge in the knots domain and if this knowledge is easily addressable and retrievable. The purpose of this system is to categorize seaman's knots by their features in a taxonomy structure and allow their retrieval based on the belief of presence or absence of features.

In this work, we focused on verifying if a hierarchical categorization was a reliable approach to represent seaman's knowledge in the knots. This system categorized seaman's knots by their features in a taxonomy structure and their retrieval based on the belief of presence or absence of features. We constructed an easy manageable and accessible domain to produce a taxonomy that was hierarchically clustered in order to find relations between knots by observing the shared features. It was built a decision tree for the nodes of the dendrogram of dimension 2k – 1 and the algorithm retrieved the most similar knots in the taxonomy comparing to those in the query. The curse of dimensionality was eased by pre-aggregating atomic features and the clusters trended to be more compact because the differentiation factors were ignored.

Even though the Ward's method gives a negative indication on the compactness of the dendrogram, we found that it is not sensible to the distance function that is used. When using Hamming distance the Sum of Squared Errors is bigger than the Euclidean distance, even when the two trees are the same. This result is acceptable, even though the two trees have relatively the same compactness.

A main contribute of this work was a novel uncertainty calculation method inspired by quantum computation. The uncertainty was measured in parallel during running time by calculating all possibilities. In the end all the results were known and had an aggregated probability.

This work created a dataset of knots that can be used in future works in the domain of seaman's knots, characterizing the knots by their features in two levels of abstraction.

We were able to produce a system that is simple, efficient, extendable, and intuitive. It is possible for us to search for an unknown domain and have results we can interpret.

This work allows a user to search for knots that both have and don't have some feature.

Future work

Domain Specific words: Change the knots' features to fit the system onto a domain-specific area, like Forensics (Budworth 1982). Identifying knots can lead to clues in a police investigation.

Expand the number of knot features: By analysing in a greater depth the Ashley Book of Knots (Ashley 1944) and other books (Cotterell & Storm 2003; Silva 1992), more features can be introduced, leading to a more detailed knots' representation.

Associate images with the knots: By associating the images to the actual knots and to their features in this system it would be easy to visualize what the features correspond to. One of the possible outcomes is the integration with CLUE (Chen et al. 2003), a content-based image retrieval that uses clustering. Plus, by adding step-by-step guides on how to tie the knots would contribute to skill acquisition for the user (Schwan & Riempp 2004).

Other domains: In theory, by following the same structure to organize the data, it is possible to change the domain. Let us say we want to categorize whisky based on its features (a whisky can be peated, fruity, sweet, salted…). There are subtleties in the flavors (fruity.apricot, sweet.vanilla, sweet.chocolat). The same principle can be applied as if they were knots.

WordNet: A possible alternative to the implemented internet solution for the synonymous retrieval is to use a local lexical database like WordNet (Miller 1995).

References

Ashley, C.W., 1944. The Ashley Book of Knots.

Berry, M.J. & Gordon S. Linoff, 2011. Data mining techniques-for marketing, sales and customer support. , p.888.

Beyer, K. et al., 1999. When is "nearest neighbor" meaningful? *Database Theory—ICDT'99*, pp.217–235.

Cotterell, A. & Storm, R., 2003. *The Ultimate Encyclopedia of Mythology* S. Duffi, ed., Barnes & Noble.

Everitt, B., 2011. Cluster analysis. *Wiley series in probability and statistics*, p.xii, 330 p.

Fournier-Viger, P., 2014. SPMF: A Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research*, 15, pp.3569–3573.

Gorunescu, F., 2011. *Data mining: Concepts, models and techniques*, Morgan Kaufmann.

Halkidi, M., Batistakis, Y. & Vazirgiannis, M., 2001. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3), pp.107–145.

Hamming, R.W., 1950. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2), pp.147–160.

Huth, A.G. et al., 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600), pp.453–458.

Jaccard, P., 1901. Etude de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37(142), pp.547–579.

Johnson, S.C., 1967. Hierarchical clustering schemes. *Psychometrika*, 32(3), pp.241–254.

Liu, B., Xia, Y. & Yu, P.S., 2000. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*. New York, New York, USA: ACM Press, pp. 20–29.

Lucas, P. & van der Gaag, L., 1991. Principles of Expert Systems. , p.426.

Novak, J.D., 2010. Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations. *Journal of E-Learning and Knowledge Society*, 6(3), pp.21–30.

Osherson, D.N., Scarborough, D. & Sternberg, S., 1998. *An Invitation to Cognitive Science* E. E. Smith & D. N. Osherson, eds., MIT Press.

Otman, G., 1996. Les Représentations Sémantiques en Terminologie. *Masson*.

Quinlan, J.R., 1986. Induction of Decision Trees. *Machine Learning*, 1(1), pp.81–106.

Russel, S. & Norvig, P., 2010. *Artificial Intelligence: A Modern Approach*,

Silverstein, S.M., 2012. In Search of an Inca: Identity and Utopia in the Andes. Alberto Flores Galindo, Edited and Translated By CarlosAguirre, Charles F.Walker, and WillieHiatt, New York: Cambridge University Press, 2010. 270 + xxix pp. *The Journal of Latin American and Caribbean Anthropology*, 17(2), pp.345–346.

Smith, J.Q., 2010. *Bayesian Decision Analysis: Principles and Practice*,

Sowa, J., 1999. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Course Technology.

Steel, N.M., 2012. *Encyclopedia of the Sciences of Learning* N. M. Steel, ed., Springer.

Stillings, N.A. et al., 1987. *Cognitive Science: An Introduction*, MIT Press.

Tversky, A., 1977. Features of similarity. *Psychological review*, 84(4), p.327.

Wichert, A., 2000. Categorical expert system `Jurassic’. *Expert Systems with Applications*, 19(3), pp.149–158.

Wichert, A., 2014. *Principles of Quantum Artificial Intelligence*, World Scientific Book.

Zhou, T. et al., 2003. Mx : A programming language for scientific computation.