Automated Taxonomy Building by Adopting Discriminant and Characteristic Capabilities

Giuliano Armano, Alessandro Giuliani, and Emanuele Tamponi

Department of Electrical and Electronic Engineering,
University of Cagliari
Via Marengo 2, 09123, Cagliari, Italy

Abstract. Taxonomies are becoming essential in several fields, playing an important role in a large number of applications, particularly for specific domains. Taxonomies provide efficient tools to people by organizing a huge amount of information into a small hierarchical structure. Taxonomies were originally built by hand, but nowadays the technology permits to produce a vast amount of information. Consequently, recent research activities have been focused on automated taxonomy generation. In this paper, we propose a novel approach for automatically build a taxonomy, starting from a set of categories. We deem that, in a hierarchical structure, each node should intuitively be represented with proper meaningful and discriminant features, instead of considering a fixed feature space. Our proposal relies on two metrics able to identify the most meaningful features. Our conjecture is that a feature could significantly change its discriminant power (hence, its role) along the taxonomy levels. Hence, we devise a greedy algorithm able to build a taxonomy by identifying the meaningful terms for each level. We perform preliminary experiments that give rise to the usefulness of the proposed approach.

1 Introduction

Taxonomies play an important role in a growing number of application, particularly for specific domains. Originally built by hand, taxonomies have been recently focused on their automatic building. In particular, a crucial issue of taxonomy building is the choice of the most suitable features (e.g., meaningful terms in textual documents). We deem that, in a hierarchical structure, a node should intuitively be identified by proper discriminant terms, rather than defining a sole feature space for the entire taxonomy.
In this paper, we define a novel approach for automatically build a taxonomy, starting from a set of categories. We adopt two novel metrics, i.e., the *discriminant capability* and the *characteristic capability* [1], the former growing in accordance with the ability to distinguish a given category against others, whereas the latter grows in accordance to how the feature is frequent and common over all categories. Our conjecture is that a feature could change its role, depending on its discriminant power, along the taxonomy levels. We assert that this behavior can be exploited for devising an automatic taxonomy building approach. In so doing, we propose an algorithm able to build taxonomies by identifying

the meaningful terms for each level. In this work, the underlying scenario is text categorization, where source items are textual documents (e.g., webpages, online news, scientific papers, or e-books), and the features are the terms in the documents.

The rest of the paper is organized as follows: Section 2 presents the background and the related work on taxonomy generation; Section 3 describes the discriminant and characteristic capabilities, whereas in Section 4 the proposed algorithm is described and detailed; experiments are reported in Section 5, while Section 6 ends the paper with the conclusions and the future work.

2 Background

Nowadays, taxonomies are indispensable to a huge number of applications. For example, in web search, organizing domain-specific queries into a hierarchy can help to better understand the queries and improve search result [13], or to improve query refinement [11]. Taxonomies, originally built by hand, have been recently focused on their automatic generation. Several works have been devoted to taxonomy induction, in particular with respect to automatically creating a domain-specific ontology or taxonomy [7–9]. Several works have been based on hierarchical clustering algorithms. In particular, there are two main approaches of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). The former regards each data item as a cluster, and clusters are recursively merged; the latter considers the entire dataset as a cluster, and then clusters are recursively split. Both approaches end when a stop criterion is yielded.
The hierarchical agglomerative clustering (HAC, hereinafter) has been widely adopted for building hierarchies, e.g., in the work of Li et al. [6], that proposes an algorithm able to build a dendrogram (basically, a binary tree). On the other hand, further works proposes divisive approaches, as in the work of Punera et al. [10]. Chuang et al. proposed a hybrid approach, in which, essentially, the binary tree obtained from HAC is modified by a divisive task in order to obtain a wide tree with multiple children [5].

An important task for taxonomy building is to recognize the most meaningful features. Our insight is that, due to the hierarchical structure, each node intuitively should be represented with proper meaningful terms, instead of considering a fixed vocabulary for the entire structure. we deem that each document collection is unique, making useful to devise methods and algorithms able to automatically build a distinct list of meaningful features for each collection.

3 The Adopted Metrics

In this Section we describe the adopted metrics and their properties.

3.1 Discriminant and Characteristic Capabilities

The metrics have been devised for both classifiers performance assessment and feature selection [1]. We apply these metrics for feature selection, as they are

able to evaluate the *discriminant* (δ) and *characteristic* (φ) capabilities of each feature.

In the underlying scenario of text categorization, δ measures the ability of a term to distinguish a given category C against others, whereas φ measures to which extent a term is pervasive in the given set of documents. in formula:

$$\delta = \frac{\#(t, C)}{\#(C)} - \frac{\#(t, \overline{C})}{\#(\overline{C})} \tag{1}$$

$$\varphi = \frac{\#(t, C)}{\#(C)} + \frac{\#(t, \overline{C})}{\#(\overline{C})} - 1 \tag{2}$$

where a generic term t contained in a document represents the *binary* feature under analysis, meaning that it can be assume two values, depending on the presence or absence in the document. The meaning of each component in the formulas are the following: $\#(t, C)$ is the number of documents of C containing t; $\#(t, \bar{C})$ is the number of documents of $\bar{C}$ containing t; $\#(C)$ is the total number of documents of C; $\#(\bar{C})$ is the total number of documents of $\bar{C}$.
Let us recall that in [1] definitions are given for a binary problem, meaning that, for a given class C, the alternate category $\bar{C}$ is the union of all the categories except C.

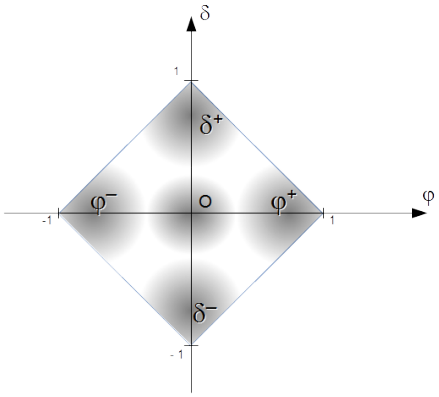

Fig. 1: The regions of the space.

3.2 Terms Roles

Assuming both ranging from -1 to +1, the metrics described by formulas 2 and 1 show an orthogonal behavior; furthermore, it has been proved that the $\varphi - \delta$ space is constrained by a rhomboidal shape [1]. In this context, a term plays a

distinct role in each category, depending on the rhombus region in which the term falls.
Important terms for text classification appear in upper and lower corner of the rhombus in Figure 1, as they have high values of $|\delta|$. In particular, a high positive value of δ (the region marked as δ^+ in the Figure 1) means that the term frequently occurs in C and is rare in $\bar{C}$; ideally, δ is $+1$ when the term occurs in all documents of C and no documents of $\bar{C}$ contain it. Conversely, a high negative value of δ (the δ^- region) means that the term frequently occurs in $\bar{C}$ and is rare in C; ideally, $\delta = -1$ means that all documents of $\bar{C}$ contain the term, and no documents of C contain it. As for the characteristic capability, terms that occur barely on the entire domain are expected to appear in the left corner of the rhombus (φ^-), while stopwords are expected to appear in the right handed corner (φ^+). Ideally, $\varphi = +1$ when the term occurs in each document of the entire domain, whereas $\varphi = -1$ when the term is completely absent in the domain. Figure 1 outlines the expected behavior for all cases.

Terms falling in φ^+ do not necessarily represent typical stopwords *only* (i.e., common articles, nouns, conjunctions, verbs, and adverbs). Rather, also domain-dependent stopwords are located in that area [3].

Moreover, theoretically, if a term has a zero value for both δ and φ in a given category C, it is equally distributed in the domain in this way: half of C documents contain the term, and also half of documents of the alternate category $\bar{C}$ contain the term. If a term is projected close to the origin of the space, there is uncertainty in considering the term as stopword, irrelevant, or discriminant[1].

In a previous work a preliminary analysis on how each feature changes its role along taxonomy nodes has been performed [2], showing that a discriminant term tends to become irrelevant when moving up in the taxonomy path.

Furthermore, a domain-dependent stopword becomes discriminant in the upper levels, giving rise to the relevance of such terms.

4 Methodology

The algorithm proposed in this work is based on a *bottom-up approach* for building a hierarchy tree, starting from a set of *leaf categories* over a corpus of documents. The information needed to initialize the algorithm is, for each category: a) the number of documents falling into the category and b) for each term t in the corpus, the number of documents containing t.

In the previous work about the term roles in a taxonomy [2] the metrics are computed by considering a binary problem, in which the alternate category has been considered as the union of all siblings of the positive node. The devised algorithm adopts a generalization to a multi-category case. After these preliminary insights, we present our algorithm for taxonomy building.

[1] An analysis of this region is a part of future work.

4.1 Identifying Characteristic and Discriminant Terms

As discussed in the previous section, the properties of a term depend on which region of the $\varphi - \delta$ rhombus it falls. Let us define two regions: the *characteristic* ($\mathcal{A}_\varphi$) and the *discriminant* ($\mathcal{A}_\delta$) areas. In this preliminary study, we simply identify the areas by the following schema:

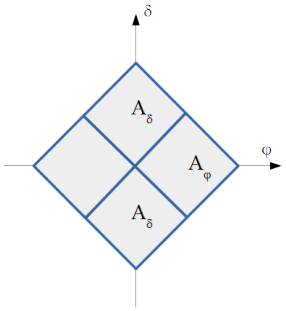

Fig. 2: The $\mathcal{A}_\varphi$ and $\mathcal{A}_\delta$ areas.

We say that a term is characteristic (discriminant) of two categories C_i and C_j if it falls in the characteristic (discriminant) area:

$$\Phi(t; C_i, C_j) = \begin{cases} 1 & (\varphi, \delta) \in \mathcal{A}_\varphi \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\Delta(t; C_i, C_j) = \begin{cases} 1 & (\varphi, \delta) \in \mathcal{A}_\delta \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

These equations need to be generalized for a hierarchy. We define a hierarchy as a series of *level* represented by a *layer* of categories. Each layer contains the categories belonging to the respective level of the hierarchy, grouped by their parent category. For example, if a particular level contains the categories A, B, C, D, E, and A, B, C have X as parent category and D, E have Y as parent category, the layer for that level will be defined as: $\{A, B, C\}, \{D, E\}$. We call $\{A, B, C\}$ and $\{D, E\}$ *siblings groups*. We can see a layer as the partition of the categories in a level, defined by their sibling relations.

We define the indicator function Φ for the *characteristic terms* of a sibling group $\mathcal{S}_k$ as follows:

$$\Phi(t; \mathcal{S}_k) = \begin{cases} 1 & \sum_{i=1}^{|\mathcal{S}_k|} \Phi(t; C_i, \overline{C}_i) \geq \dfrac{|\mathcal{S}_k|}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\bar{C}_i$ is the alternate category of a given class C_i, i.e., the union of the siblings of C_i.

In a similar way we can define the indicator function Δ for the *discriminant terms* of a sibling group in a layer $\mathcal{L}$:

$$\Delta(t; \mathcal{S}_k, \mathcal{L}) = \Delta(t; P(\mathcal{S}_k), P(\overline{\mathcal{S}_k})) \tag{6}$$

where $\bar{S}_k$ is the union of the rest of siblings groups, and $P(S)$ indicates the parent node for a siblings group S.

Using Eq. 5 and Eq. 6 we can now define the *set of characteristic terms* $\mathcal{T}_\varphi$ and the *set of discriminant terms* $\mathcal{T}_\delta$ for each siblings group in a layer:

$$\mathcal{T}_\varphi(\mathcal{S}_k) = \{t : \Phi(t; \mathcal{S}_k) = 1\} \tag{7}$$

$$\mathcal{T}_\delta(\mathcal{S}_k, \mathcal{L}) = \{t : \Delta(t; \mathcal{S}_k, \mathcal{L}) = 1\} \tag{8}$$

4.2 Identifying the Optimal Layer

The proposed bottom-up algorithm is split into a series of optimal partition problems. For the current set of categories at hand (starting from the leaves), we strive for identifying the *optimal layer*, that is, the optimal set of siblings groups. Once these groups are identified, the categories inside them are "collapsed" together to generate the parent categories, and the algorithm goes on recursively, layer by layer, until some stop condition is met.

To identify the optimal layer we use a target function that is derived from Eq. 7 and Eq. 8. The rationale behind this function has already been discussed in Sec. 3.2: the behavior we expect in a taxonomy is that a large number of characteristic terms of a sibling group become determinant terms of the parent category. We define the target function, that we call *layer score* (G), as following:

$$G(\mathcal{L})_B = \sum_{k=1}^{|\mathcal{L}|} |\mathcal{T}_\varphi(\mathcal{S}_k) \cap \mathcal{T}_\delta(\mathcal{S}_k, \mathcal{L})| \tag{9}$$

The optimization task that identifies the optimal layer $\mathcal{L}^*$ can then be stated as follows:

$$\mathcal{L}^* = \arg\max_{\mathcal{L}} G(\mathcal{L}) \tag{10}$$

Solving exactly Eq. 10 is an NP-hard problem as the search space is defined by the Bell number [4]. We propose a simple greedy algorithm, and we will show in the Experimental section that the solutions it provides are good enough for a wide range of taxonomies.

The algorithm starts from the trivial layer, in which each siblings group contains exactly one category, and proceeds recursively. The layer under analysis is the *current candidate*. At each step, the algorithm looks for all the layers reachable by moving exactly one category from its current sibling group to another group. The algorithm then evaluates the layer score of each of these layers, and if the maximum score exceeds the score of the starting layer, the associated layer

becomes the current candidate. If not, the current candidate is chosen as the one satisfying (approximately) Eq. 10.

Once the optimal layer has been chosen, each sibling group is collapsed to generate the *parent categories* (P):

$$P(\mathcal{S}_k) = \bigcup_{i=1}^{|\mathcal{S}_k|} C_i \tag{11}$$

In the rest of the discussion, it is implicit that some sort of mechanism is in place to keep track of which categories (if any) are children of a given category.

4.3 Growing the Taxonomy

We are now able to describe the entire algorithm succinctly. It expects the following inputs:

- the set of leaf categories $\mathcal{C}_{\text{leaves}}$;
- for each leaf category C_i, all the data needed to calculate φ and δ, described in Sec. 3.1.

The output is the generated taxonomy T. The algorithm then proceeds iteratively, until the stop condition (Step 6) is met. There are two distinct loops: in the outer one, we build the taxonomy one level at a time; in the inner loop, we search for an approximation of the optimal layer.

1. Add the current set of categories to the taxonomy T as a *taxonomy level*;
2. Put the current set of categories in a *trivial layer*;
3. Mark the trivial layer as the chosen layer $\mathcal{L}_c$ and calculate its score G_c;
4. Mark the *next layer* $\mathcal{L}_n = \mathcal{L}_c$, and *next layer score* $G_n = G_c$;
5. For each (unordered) pair of sibling groups in $\mathcal{L}_c$:
   (a) Evaluate the score G_h of the layer obtained by merging the pair of groups, $\mathcal{L}_h$;
   (b) If $G_h > G_n$: set $\mathcal{L}_n = \mathcal{L}_h$ and $G_n = G_h$;
6. If $\mathcal{L}_n$ is still the trivial layer: the stop condition is met: return T.
7. If $\mathcal{L}_n \neq \mathcal{L}_c$: set $\mathcal{L}_c = \mathcal{L}_n$ and $G_c = G_n$; go back to Step 5;
8. Otherwise: consider $\mathcal{L}_c$ the optimal layer and collapse it to generate the set of parent categories (use Eq. 11 on each of its sibling groups); go back to Step 1.

5 Experiments

Assessing an algorithm for the generation of taxonomies is an hard task. This is due to the fact that existing taxonomies are far from being considered "golden standards", that is, they are not precise enough to guarantee that they can be taken as absolute reference during the test phase. Moreover, many metrics exist to measure the agreement between two taxonomies, and none of them is universally accepted as a good one.

5.1 Assessing the Learned Taxonomy

Let T_R be the *reference taxonomy* and T_L the *learned taxonomy*, that the algorithm described in Sec. 4.3 generates. We need to assess the learned taxonomy by comparing them with the reference.

To measure the agreement between T_R and T_L, we use the metric defined by Navigli [12]. Let $\mathcal{C}_{\text{leaves}}$ be the leaves of both the reference taxonomy and the learned one (the two sets are identical as the set of leaves is one of the inputs of the taxonomy generation algorithm). Let k be the depth of the reference taxonomy. If the depth of the learned taxonomy is at least k, then for each $i \in 0, \ldots, k$ we have two *unwrapped layers*, $T_R i$ and T_L^i. An *unwrapped layer* is the same as the layer defined in the previous section, but it is always defined in terms of the leaf categories. That is, also the grouping defined by intermediate layer is not defined in term of some parent category generated by collapsing its children, but from the children themselves; we call them unwrapped layer as we are effectively unwrapping each collapsed sibling group up to the leaves. By definition, $T_R^0 = T_L^0 = \{\mathcal{C}_{\text{leaves}}\}$, that is, the root layer contain a single group that has all the leaves.

To assess the agreement between the two taxonomies, we first measure the agreement between two unwrapped layers:

$$B^i = \frac{n_{11}^i}{\sqrt{(n_{11}^i + n_{10}^i) \cdot (n_{11}^i + n_{01}^i)}} \tag{12}$$

Where n_{11} is the number of pair of leaves included in the same group in both layers, n_{01} is the number of pair of leaves included in the same group in the learned layer but not in the reference layer, and viceversa for n_{10}.

The overall metric is then defined as:

$$B(T_R, T_L) = \frac{2}{k+1} \sum_{i=0}^{k-1} \frac{i+1}{k} B^i \tag{13}$$

It can easily be noticed that B takes into higher consideration the layer metric of the deeper layers in the taxonomy. This is necessary to counterbalance the fact that as less sibling groups are found in the first layers, the probability that a pair of categories is found in the same group just by chance is increased; they thus need to be given a lower weight in the overall metric.

If the depth of the learned taxonomy is different from the depth of the reference one, we have to slightly adjust the previous definition:

- if $k_L > k_R$: the layers after k_R are ignored: this way we give an higher score to more structured learned taxonomies;
- if $k_L < k_R$: the last layer in the learned taxonomy is repeated as many times as needed to reach k_R. This way less structured taxonomies get a lower score.

Table 1: Taxonomies for the experiments. They where built starting from the DMOZ webpage taxonomy, and organized to have different degree of difficulty.

Name	Leaves	Base	Intermediate
easy1	10	2	1
easy2	6	2	1
easy3	10	2	1
easy4	9	2	1
easy-medium1	12	3	1
easy-medium2	10	3	1
easy-medium3	15	3	1
medium1	8	1	2
medium2	9	1	2
medium3	4	1	2
medium4	8	1	2
medium5	12	1	2
medium-hard1	12	2	4
medium-hard2	15	2	4
medium-hard3	25	2	4
hard1	13	1	3
hard2	8	1	3
hard3	15	1	3
hard4	17	1	3

5.2 Reference Taxonomies

Experiments are performed using a collection of webpage documents. The dataset is extracted from the DMOZ taxonomy[2]. Aside from the leaves, each node is built with the union of the children's documents. Textual information from each page code is extracted, and each document is converted into a bag of words representation, each word being weighted with two values: δ and φ, computed by applying equations 1 and 2. We built 19 small sub-taxonomies of DMOZ. We grouped them in terms of "difficulty". Table 1 shows the main properties of each taxonomy. "Easy" and "easy-medium" taxonomies have base categories very different between them, so also the leaves will clearly be very different *or* very similar between them, and the grouping task should be easier. "Medium" taxonomies stem from the same root category, so the similarity between leaves increases. Finally, "medium-hard" and "hard" taxonomies have a lot more leaves and multiple root and intermediate categories. Fig. 3 and Fig. 4 are examples of taxonomies used in the experiments.

5.3 Results

Tab. 2 contains a summary of the experimental results, and Fig. 5 shows an example of taxonomy learned by the algorithm. Let us note that the generated

[2] http://www.dmoz.org

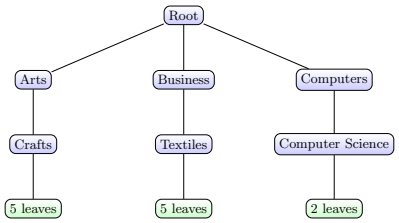

Fig. 3: *easy-medium1* taxonomy. You can notice how each group of leaves is "distant" from the others. High cohesion of groups and high discriminant characteristic let us suppose that this kind of hierarchy is easy to learn.

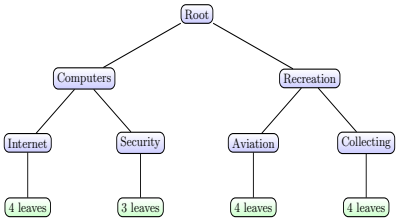

Fig. 4: *medium-hard2* taxonomy. In this case the taxonomy is far more complex than that shown in Fig. 3. We expect that the algorithm will make more errors on this kind of taxonomy.

Table 2: Summary of the experimental results. We show both the per-layer score and the overall taxonomy score.

Name	B^1	B^2	B
easy1	0.400	–	0.600
easy2	0.309	–	0.539
easy3	0.429	–	0.619
easy4	0.533	–	0.689
easy-medium1	0.636	–	0.757
easy-medium2	0.787	–	0.858
easy-medium3	0.707	–	0.805
medium1	0.655	0.686	0.728
medium2	0.667	0.553	0.666
medium3	0.577	0.0(!)	0.359
medium4	0.655	0.577	0.674
medium5	0.674	0.511	0.647
medium-hard1	0.438	0.735	0.680
medium-hard2	0.677	0.436	0.610
medium-hard3	0.589	0.229	0.477
hard1	0.734	0.446	0.634
hard2	0.756	0.500	0.669
hard3	0.931	0.0(!)	0.477
hard4	0.697	0.295	0.546

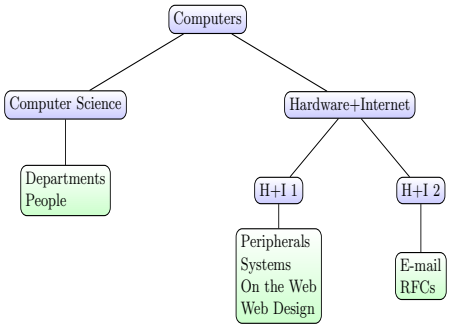

Fig. 5: Learned taxonomy starting from the leaves in the "hard2" taxonomy. The *Computer Science* category has been correctly split from the other categories in the same levels, whereas the categories *Hardware* and *Internet* were first grouped together and then sub-split in a wrong way: *On the Web* and *Web Design* should be put along with *E-mail* and *RFCs*. The total score of the learned taxonomy is $B = 0.669$.

taxonomy is very close to the original one even from a qualitative point of view. The obtained scores confirm the quality in most of the cases.

6 Conclusions and Future Work

In this paper, a novel approach for automatically build a taxonomy has been proposed. The proposal adopts two novel metrics, i.e., the *discriminant capability* and the *characteristic capability*, able to measure the discriminant power of a feature. The former grows in accordance with the ability to distinguish a given category against others, the latter grows in accordance to how the feature is frequent and common over all categories. Our conjecture is that a feature could change its role, depending on its discriminant power, along the taxonomy levels. This behavior has been exploited for devising the algorithm, that is based on the identification of the meaningful terms for each level. In particular, we devise a greedy bottom-up algorithm that recursively identifies the optimal layer for each level.

The proposal, even if it is still in a preliminary stage, provides encouraging results, as shown by the experiments. As for future work, we are currently improving and refining the algorithm, with different strategies. Furthermore, a set of comparing experiments with several state of the art approaches are planned, in order to give rise to the usefulness of the proposed approach.

References

1. Armano, G.: A direct measure of discriminant and characteristic capability for classifier building and assessment. Information Sciences 325, 466 – 483 (2015), http://www.sciencedirect.com/science/article/pii/S0020025515005241

2. Armano, G., Fanni, F., Giuliani, A.: Analysis of term roles along taxonomy nodes by adopting discriminant and characteristic capabilities. In: Proceedings of the 6th Italian Information Retrieval Workshop, Cagliari, Italy, May 25-26, 2015. (2015), http://ceur-ws.org/Vol-1404/paper_24.pdf
3. Armano, G., Fanni, F., Giuliani, A.: Stopwords identification by means of characteristic and discriminant analysis. In: Loiseau, S., Filipe, J., Duval, B., Van Den Herik, J. (eds.) 7th International Conference on Agents and Artificial Intelligence 2015 (ICAART 2015). pp. 353–360. SCITEPRESS Science and Technology Publications, Lisbon, Portugal (10–12 Jan 2015)
4. Bell, E.T.: The iterated exponential integers. Annals of Mathematics 39(3), 539–557 (1938), http://www.jstor.org/stable/1968633
5. Chuang, S.L., Chien, L.F.: A practical web-based approach to generating topic hierarchy for text segments. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management. pp. 127–136. CIKM '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1031171.1031193
6. Li, T., Zhu, S., Ogihara, M.: Hierarchical document classification using automatically generated hierarchy. Journal of Intelligent Information Systems 29(2), 211–230 (2007), http://dx.doi.org/10.1007/s10844-006-0019-7
7. Mani, I.: Automatically inducing ontologies from corpora. In: Proceedings of CompuTerm 2004: 3rd International Workshop on Computational Terminology, COLING'2004 (2002)
8. Navigli, R., Velardi, P., Faralli, S.: A graph-based algorithm for inducing lexical taxonomies from scratch. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three. pp. 1872–1877. IJCAI'11, AAAI Press (2011), http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-313
9. Poon, H., Domingos, P.: Unsupervised ontology induction from text. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 296–305. ACL '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), http://dl.acm.org/citation.cfm?id=1858681.1858712
10. Punera, K., Rajan, S., Ghosh, J.: Automatically learning document taxonomies for hierarchical classification. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web. pp. 1010–1011. WWW '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1062745.1062843
11. Sadikov, E., Madhavan, J., Wang, L., Halevy, A.: Clustering query refinements by user intent. In: Proceedings of the 19th international conference on World wide web. pp. 841–850. WWW '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1772690.1772776
12. Velardi, P., Faralli, S., Navigli, R.: Ontolearn reloaded: A graph-based algorithm for taxonomy induction. Computational Linguistics 39(3), 665–707 (2013), http://dblp.uni-trier.de/db/journals/coling/coling39.html#VelardiFN13
13. White, R.W., Bennett, P.N., Dumais, S.T.: Predicting short-term interests using activity-based search context. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 1009–1018. CIKM '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1871437.1871565