# Active Semi-Supervised Concept Hierarchy Refinement

**Michał Dereziński** MICHAL.DEREZINSKI@GMAIL.COM
*Faculty of Mathematics, Informatics and Mechanics,*
*University of Warsaw, Banacha 2, 02-097 Warsaw, Poland*

## Abstract

Concept hierarchy generation has multiple applications, including creating ontologies and data mining. When performed manually, it is a very time consuming task. On the other hand, unsupervised techniques often provide very limited or inaccurate results. There have been some attempts to address it with semi-supervised methods, however they still require significant human effort and are mostly limited to simple cluster hierarchies. This paper presents a new algorithm for active semi-supervised concept hierarchy refinement, which can be used together with any unsupervised method. We propose a very simple query format, which can be answered by a human expert with minimal effort. Moreover, an active learning strategy is presented for improving the results. The algorithm is tested in conjunction with two concept hierarchy generation techniques. The results show great improvement of accuracy even with small human effort and a significant benefit from utilizing active learning.

**Keywords:** Concept Hierarchy, Information Retrieval, Semi-supervised, Active Learning, Clustering, Formal Concept Analysis

## 1. Introduction

Generating concept hierarchies has many applications in areas such as Information Retrieval, Natural Language Processing or Data Mining. Grouping data elements into sets with a hierarchical structure helps visualising large amounts of information and analyzing them with varying levels of detail. Particularly in Natural Language Processing, concept hierarchies play a vital role in creating ontologies, like Wordnet. The size of such structures makes manual generation impractical or even impossible. However, current methods for automatic generation of concept hierarchies are often unable to achieve proper accuracy for complex concept structures.

Therefore, it is important to consider semi-supervised algorithms for generating concept hierarchies, which combine the benefits of human expertise with the efficiency of computers. The standard approach to semi-supervised learning is based on a classification task. The training set consists of some number of labeled data, and also a large set of unlabeled elements. By combining information from both labeled and unlabeled data points, we expect to get a better classifier than if we ignored the unlabeled data, as described in (Blum & Mitchell, 1998). Although the task of generating a concept hierarchy is not a classification problem, it is possible to provide supervised information that can be used to improve the results.

One of the goals of semi-supervised learning is to minimize the work that needs to be spent by human experts labeling data for the algorithm. For example, it can be reduced

by active learning systems, which intelligently choose those data points that should be presented to the experts and those that should be left unlabeled. This often speeds up the learning, when compared to selecting random data for labeling.

In this paper, we present an active semi-supervised method of improving automatically generated concept hierarchies. Our approach can be applied alongside virtually any algorithm for generating concept hierarchies. It also supports not just tree hierarchies, but also concept lattices, as well as any other directed acyclic structures based on the subset relation. First, we describe the format of questions that are presented to the human expert. This process is designed to be as effortless as possible, so that a large amount of data can be labeled in a short period of time. For each question, the expert is only required to look at three data elements and decide which one is most different from the other two. In many domains this task can be completed in just a few seconds. We also describe an algorithm that uses the obtained information to refine the hierarchy of concepts. Finally, we present a method of actively choosing a specific query, based on the hierarchy that was constructed at that point. For evaluation, we created a system which generates a concept hierarchy from fuzzy data, and then refines it through a sequence of queries. The queries are automatically answered using a manually generated hierarchy. That hierarchy is also used to measure performance of the algorithm.

## 2. Related Work

Unsupervised concept hierarchy generation has been widely studied in various forms. One of the approaches to this task is Formal Concept Analysis (FCA), described in (Ganter & Wille, 1999), and its derivations, like Fuzzy and Fault Tolerant FCA, presented in (Tho et al., 2006) and (Pensa & Boulicaut, 2005) respectively. Another method related to this approach is Fuzzy Norris' algorithm, which was introduced in (Dereziński, 2012). We will use it here to provide the initial concept hierarchy for evaluation.

The discussed problem can also be viewed as a clustering task. The most widely used clustering algorithm is K-Means, for example in (Basu et al., 2002). However, this and most other methods produce flat structures which are unsuitable for presenting concept hierarchies. Hierarchical Agglomerative Clustering is the main clustering technique which provides a structure appropriate for representing concepts. However, the hierarchies it produces are limited to a tree structure which is insufficient for some tasks. This method, along with others, is discussed in (Grira et al., 2005).

There are several papers discussing semi-supervised approaches to clustering. For example, (Basu et al., 2002) proposes using a small set of labeled data points as a seed for clustering other unlabeled data. However, it returns a flat set of clusters. Other semi-supervised clustering algorithms often use pair-wise constraints, which allow experts to specify that some pairs of objects should or should not belong to the same cluster. This approach is also limited to flat concept structures. On the other hand, (Bade & Nürnberger, 2008) proposes a hierarchical clustering model where a subset of objects manually converted into a hierarchy is used to create constraints for the final algorithm. The triple-wise constraints discussed there are similar to the queries that we propose in this paper. A completely different approach is presented in (Mao et al., 2012), where a hierarchical topic model is extracted from text documents through Semi-Supervised Hierarchical Latent Dirichlet Allocation. Here,

the authors assume that the documents are manually annotated with information that is be relevant for generating a topic hierarchy. However, both of the approaches, however, do not allow for active learning in the process of constraint selection or annotation. They also potentially require considerable effort from human experts.

Recently, there have been some attempts at employing active learning for clustering, however most deal with non-hierarchical structures (Basu et al., 2002; Grira et al., 2005). An interesting way of combining Hierarchical Agglomerative Clustering with active learning is presented in (Nogueira et al., 2011). However, to achieve noticeable improvement from supervised information, this system has to ask human experts fairly complex queries.

Additionally, all of the mentioned methods integrate semi-supervised information directly into the clutering algorithms and are limited to tree hierarchies, while our approach can be applied to any hierarchy of clusters or concepts. Therefore, it is much more flexible and has wider use case.

## 3. Active Concept Hierarchy Refinement

In this section we present the Active Concept Hierarchy Refinement algorithm.

### 3.1. Problem Definition

The goal of concept hierarchy generation is, given a set $O$ of objects, create a family of subsets of $O$, which together with the relation of being a subset produces a concept hierarchy. We assume that a concept hierarchy has been automatically generated by some algorithm that is based on a dataset of objects. Now the task is to refine it with the help of a human expert. We expect that the expert provides information based on desired properties of the perfect hierarchy. Those properties include the size and granularity of the structure as well as choice of the concepts themselves. The answers do not need to be always correct, because the algorithm should adjust to occasional mistakes or inconsistencies.

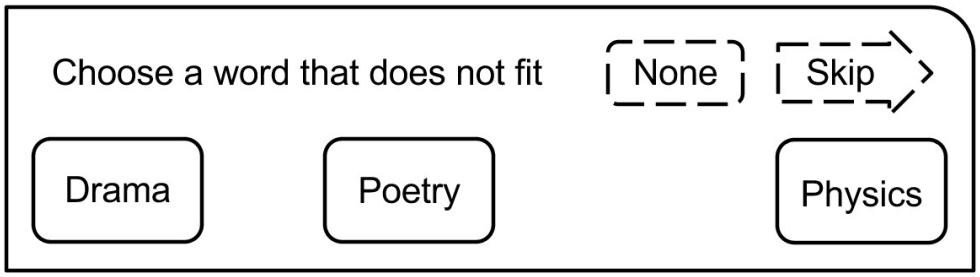
### 3.2. Interaction with Expert

The system interacts with expert through a series of actively generated queries. The queries should be easily presentable. Therefore, the information should be represented using descriptive language or images rather than a series of numbers. We propose a query which consists of a triple of objects $o_1, o_2, o_3 \in O$. The user may give one of the following responses:

1. pick one object which does not fit to the other two,
2. say that it is hard to choose such an object,
3. or claim insufficient knowledge to provide the answer.

In practical applications a crucial aspect of this process is creating a graphical user interface which makes the question answering as efficient as possible. We will not be discussing this in detail. However, it is worth noting that such a system could provide a visual hint for the user, based on a prediction from current concept hierarchy. Such an example is shown in Fig. 1 with a query for the task of categorizing document topics.

Figure 1: An example user interface for querying human expert. By properly distancing three objects the system suggests the correct answer based on its own prediction.




Let us now give a precise meaning of the expert's answer, to see how it will be understood by the system. To do that, we need to formalize a selection of the *smallest* or *largest* concept from a set of concepts. These expressions will be based on a total relation between concepts, which extends the natural subset relation. Various choices can be made for such a relation. For example, we can compare sizes of concepts – the strategy which was used during performance evaluation – or calculate the minimum top-down level of each concept in the hierarchy and use it for the comparison. From now, I will assume that such a relation has been chosen consistently for the entire algorithm. Note that it may happen that a set has multiple smallest concepts, in which case each choice of the smallest element is non-deterministic (similarly for largest elements). However, there can only be one strictly smallest element (if a set has several smallest elements, then it does not have a strictly smallest one).

**Definition 1** *Let $H \subset \mathcal{P}(O)$ be a concept hierarchy and $o_1, o_2, o_3 \in O$ be three different objects. Define* $\mathrm{Compare}(H, o_1, o_2, o_3) = o_c$ *as follows. If there exists $i \in \{1, 2, 3\}$ such that $o_i \notin C$ for every $C \in H$, then we say that $o_c :=$* Unknown*. Otherwise, for $i = 1, 2, 3$, let $C_i \in H$ be the smallest concept such that $o_i \notin C_i$, but the other two objects $o_j, o_k \in C_i$. Of course $C_i$ may not exist, so in that case we can mark it as* None*.*

*Now, there are two cases:*

1. *If all three $C_i$ are* None*, then we say that $o_c :=$* None*,*
2. *Otherwise, let $C_i$ be the strictly smallest out of the found concepts. Then, we have $o_c := o_i$. In the rare scenario, when we have at least two smallest concepts $C_i$ and $C_j$, we again set: $o_c :=$* None*.*

In other words, $\mathrm{Compare}(H, o_1, o_2, o_3)$ tries to find the smallest concept separating two of the objects from the third, and then returns that other object which *does not fit to the other two*. In some cases, particularly when all three pairs of objects have the same smallest common concept, the non-fitting object cannot be determined (answer is None).

For regular tree hierarchies (like those obtained by agglomerative clustering), this definition is significantly simpler and corresponds to the three-wise constraint definition discussed in (Bade & Nürnberger, 2008). Notice that the result is Unknown only if the hierarchy is not complete (some objects are not represented by any concepts). This is equivalent to

expert with insufficient knowledge, which may sometimes be the case. The queries for which expert returns Unknown are ignored. This information may however be relevant if there are several experts with knowledge in different subdomains. In such a case the presented method can be naturally extended to automatically detect the areas of knowledge of each expert based on when they answered Unknown. This technique is based on a broader methodology, referred to as proactive learning, which is discussed in (Donmez & Carbonell, 2008). However, we do not analyze it in this paper.

### 3.3. Main Algorithm

Listing 1 presents general schema of the refinement algorithm. Here, `GenerateHierarchy` refers to an unsupervised algorithm for generating the initial concepts. Any of the methods discussed earlier would be suitable for this task. A natural assumption is made that the algorithm includes all the relevant objects in a single top concept of the initial hierarchy. Choosing the query is performed in `ActiveSelect`. As we will show in the evaluation, random selection of three different objects is sufficient to obtain a noticeable improvement of the hierarchy. However, there is also clear benefit from utilizing the information contained within the structure of the generated hierarchy. Finally, the `Refine` algorithm will be described in the following section.

```
HIERARCHY <− GenerateHierarchy(Objects)
Expert    <− EstablishConnectionWithExpert()
repeat N times:
    Triple    <− (o1, o2, o3) <− ActiveSelect(HIERARCHY)
    Predicted <− Compare(HIERARCHY, Triple)
    Correct   <− Compare(Expert, Triple)
    if Correct not Unknown and not equal to Predicted:
        HIERARCHY <− Refine(HIERARCHY, Triple, Predicted, Correct)
```

Listing 1: Main algorithm for Active Concept Hierarchy Refinement

### 3.4. Concept Hierarchy Refinement

The proposed concept hierarchy refinement algorithm is based on the assumption that the initially generated concepts mostly describe the object domain in a reasonable way. Therefore, we can determine those concepts that do not satisfy the expert's constraint. Listing 2 presents the pseudocode of the algorithm.

The refinement algorithm assumes that `Predicted` is not equal to `Correct`. Then, if `Predicted` is not None, we can guess that similarity between two other objects from the triple is too large in the generated hierarchy. We reduce it by removing the smallest concept connecting them. The fact that such concept always exists is obvious from the definition of `Compare`.

Moreover, if `Correct` is not None, then the two other objects from the triple should be more closely related within the concept structure. So, we create a concept which better separates those two objects from the `Correct`. Note that we do this within a hierarchy

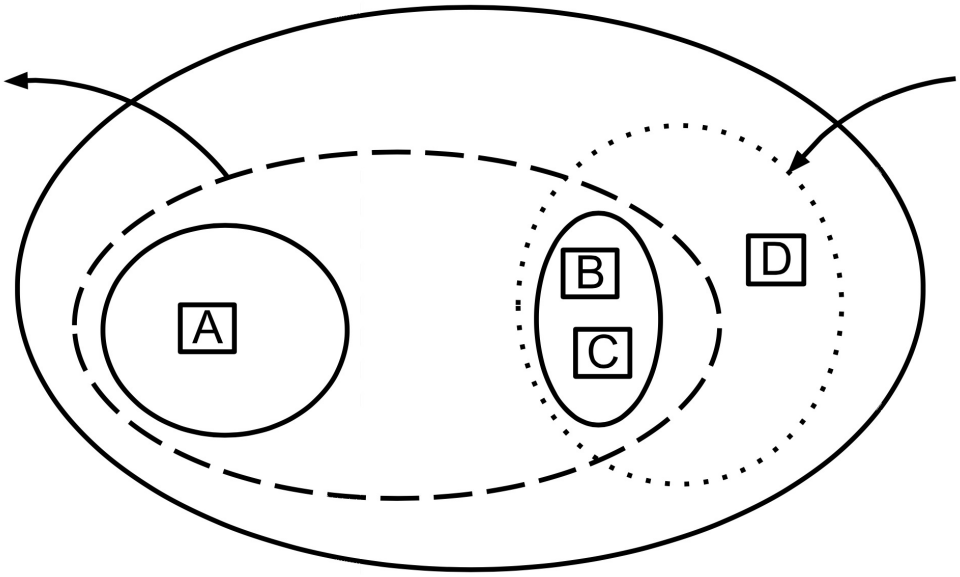
bounded to the smallest concept containing the `Triple`, because that is most likely the context in which human expert would perceive the three objects. Function `Bounded` returns this narrower hierarchy by intersecting every concept from the complete hierarchy with the bounding concept. For regular tree hierarchies, this step is unnecessary. If we do not find `Concept1` or `Concept2`, we replace them with singleton concepts, as shown in the pseudocode.

```
def Refine(HIERARCHY, Triple, Predicted, Correct):
    if Predicted not None:
        (o1, o2)  <− Triple without Predicted
        Concept   <− SmallestConcept(in = HIERARCHY,
                                     with = [o1, o2],
                                     without = Predicted)
        HIERARCHY <− RemoveConcept(HIERARCHY, Concept)
    if Correct not None:
        (o1, o2)  <− Triple without Correct
        Concept   <− SmallestConcept(in = HIERARCHY,
                                     with = [o1, o2],
                                     without = Correct)
        if Concept is None:
            Concept0 <− SmallestConcept(in = HIERARCHY,
                                        with = Triple)
            BOUNDED <− Bounded(HIERARCHY, Concept0)
            Concept1 <− LargestConcept(in = BOUNDED,
                                       with = o1,
                                       without = [o2, Correct])
                        or else Singleton(o1)
            Concept2 <− LargestConcept(in = BOUNDED,
                                       with = o2,
                                       without = [o1, Correct])
                        or else Singleton(o2)
            Concept   <− SumOfConcepts(Concept1, Concept2)
            HIERARCHY <− InsertConcept(HIERARCHY, Concept)
    return HIERARCHY
```

Listing 2: The refinement procedure executed after each query

A sample refinement operation has been presented on Fig. 2. Note that a single step of the algorithm may increase or decrease the total number of concepts by 1. In the following section we will discuss how the algorithm can be adjusted to gradually reduce the size of a hierarchy over multiple refinement steps.

Figure 2: Assume we have a hierarchy of concepts {A}, {B,C}, {A,B,C}, {A,B,C,D}. For query (A,B,D), expert answered A, while predicted answer is D. So, hierarchy refinement algorithm would remove concept {A,B,C} and add concept {B,C,D}.




### 3.5. Active Learning

Choosing which triple of objects to query during each step of the algorithm is done in function `ActiveSelect`. The simplest approach - selecting three different random objects from the entire dataset - was used as a baseline for evaluating active learning in the tests.

There are several general strategies of selecting samples to be considered. Some of the most important are uncertainty based sampling, density sampling and maximal diversity sampling. None of them is universally better than the other, so they should be chosen directly for a specific task. Often, it is optimal to alternate between two or more of those strategies. We will now discuss some of those strategies with respect to the analyzed algorithm. Since it is not a typical active learning task, there are various ways in which they can be interpreted.

#### 3.5.1. Uncertainty Sampling

For classification tasks, uncertainty sampling focuses on elements for which the classifier has lowest confidence of prediction. In our task, there is no immediately available measure to use for this strategy. Two aproaches seem worth considering:

1. It can often be possible to obtain uncertainty level for each concept generated initially by the unsupervised algorithm. For example, in (Nogueira et al. (2011)) a merge confidence is calculated for hierarchical agglomerative clustering algorithm. This can be used to determine from which concept we should select the three objects.

2. Another natural approach is to select only triples with low confidence of predicted answer to the query. This could be calculated based on the defining properties of objects, specific to a given model.

#### 3.5.2. Density Sampling

By choosing a sample from a high density cluster, we expect to improve the results of a large number of other data points that are in close proximity. On the other hand, random selection of three objects should typically result in fairly unrelated elements. It means that they will usually be connected by relatively high level concepts. This is not necessarlly a bad thing, but to balance it out, we can also encourage the selection of closely related objects. A simple way of achieving that is by picking them from one small concept. We propose an active learning strategy of first selecting a random concept and then selecting three objects from it. This encourages more densely distributed triples, without disqualifying any possible samples. Taking into account the exact object model used for initial hierarchy generation for calculating density is also worth considering.

#### 3.5.3. Maximal Diversity Sampling

This active learning strategy focuses on outliers - data points which are separated from large groups of other points. So, by contrast to density sampling, when looking for maximal diversity we want the three selected objects to only be connected by high level concepts. This sampling strategy is not typically used by itself, but as an addition to either density or uncertainty sampling. For uniformly distributed object space random selection is a reasonable way of achieving sample diversity. In the opposite case, it may be sensible to provide a more sophisticated strategy.

#### 3.5.4. Active Size Reduction

Size and granularity are a significant factor in evaluating concept hierarchies. Most unsupervised algorithms allow control of the size of the generated structure through parameters. The question is what is the optimal size of the initial hierarchy for the use with the Active Refinement algorithm. Since it mostly produces concepts from the existing ones, it is reasonable to generate a hierarchy, which is noticeably larger than the desired size. The querying process provides information about the level of concept granularity desired by the expert. When the person does not separate any one object from a triple it gives us some information about how much differentiation is required to warrant a distinct concept. Through this process, Active Refinement may somewhat reduce the size of initial hierarchy. To enhance this process, we can actively filter out some of the samples that may cause refining to enlarge the hierarchy. Specifically, whenever the predicted answer to the query is None, no concept will be removed during refinement, but one may be added. By filtering out certain percentage of such queries we can reduce the size of the hierarchy, potentially improving its measured accuracy.

## 4. Performance Evaluation

For evaluating performance of the discussed algorithm we took a set of 462 documents categorized to 77 topics which were used as labels for a topic assignment algorithm proposed and tested in (Chwiałkowski, 2012). Through that topic assignment algorithm, each label was described by a probabilistic distribution of 80 abstract topics based on word frequencies using Latent Dirichlet Allocation. This data was provided to two unsupervised concept hierarchy generating algorithms to create initial hierarchies. Additionally, a concept hierarchy was manually created which consists of 29 concepts. It was used for comparison with generated hierarchies as well as for automatic calculation of expert answers.

### 4.1. Unsupervised Algorithms

We used two different algorithms for generating initial concept hierarchies to see how the results would vary depending on what unsupervised approach was used.

#### 4.1.1. Fuzzy Norris' Algorithm

This algorithm was first proposed in (Dereziński, 2012). It is an extension of Norris' algorithm for generating concept lattices (introduced in (Norris, 1978). Beyond analyzing the strict formal concept model as the original version, it can also process fuzzy data. Therefore, it is suitable for analyzing the probabilistic distributions in our topic set. The output is a concept hierarchy similar in structure to a concept lattice. The algorithm takes one parameter - *tolerance* - which specifies the size and granularity of generated hierarchy. If it is 0, then we get a large hierarchy, potentially exponentially larger than the number of input objects. Increasing *tolerance* (which can be between 0 and 1) encourages more separated concepts and a much smaller hierarchy.

#### 4.1.2. Fuzzy Formal Concept Analysis

This formal model allows to apply Formal Concept Analysis to fuzzy data and also provides an algorithm for reducing complexity of the full concept lattice. To generate a primary hierarchy, a cut-off threshold is used to discretize the data and perform standard Formal Concept Analysis. Then, a second threshold is used to reduce the number of concepts through clustering. By adjusting second threshold from 0 to 1, we can control the size and granularity of the resulting concept hierarchy. Due to large size of the initial concept lattice, it was necessary to perform clustering several times on smaller subdomains for the tests to fit into reasonable time constraints.

### 4.2. Similarity Measure

To evaluate our results we needed to compare the similarity between two concept hierarchies. The *common information* measure defined in (Dereziński, 2012) provides such a tool. Moreover, it provides the equivalents of standard measures Precision and Recall used widely in classification problems. We have provided its definition below.

**Definition 2** *The **similarity** between sets $S$ and $T$ is defined as*

$$\mathrm{E}(S,T) = \frac{|S \cap T|}{|S \cup T|},$$

*if $|S \cup T| \neq 0$, otherwise $\mathrm{E}(S,T) = 1$. We use $|\cdot|$ to denote the size of a given set.*

**Definition 3** *Assume that we want to compare a concept hierarchy $K$ to another one, $L$. Then, **common information** between $K$ and $L$ is defined as:*

$$I(K,L) = \max_{\sigma} \sum_{(c,d)\in\sigma} E(c,d),$$

*where $\sigma$ represents any maximal one-to-one correspondence (bijective relation) between concepts from $K$ and $L$.*

Note that $I(K,K) = |K|$, where $|K|$ denotes the number of concepts in $K$. It means that $K$ provides complete information about all of its own concepts (which is a trivial statement). However, if we defined $K'$ as a hierarchy identical to $K$ and then modified one of its concepts (for example, by adding or subtracting objects from it), we would get $I(K,K') \in [|K|-1, |K|]$.

**Definition 4** *Using the above property we can define:*

$$\mathrm{Recall}(K,L) = \frac{I(K,L)}{|L|}$$

*and*

$$\mathrm{Precision}(K,L) = \frac{I(K,L)}{|K|}.$$

*Finally, we can use the standard definition for F-measure:*

$$\text{F-measure}(K,L) = 2 \cdot \frac{\mathrm{Precision}(K,L) \cdot \mathrm{Recall}(K,L)}{\mathrm{Precision}(K,L) + \mathrm{Recall}(K,L)}.$$

Table 1: Precision, Recall and F-measure for initial concept hierarchies.

| | Fuzzy Norris | | Fuzzy FCA | |
|---|---|---|---|---|
| Size | 33 | 48 | 27 | 61 |
| Precision | 0.353 | 0.271 | 0.253 | 0.152 |
| Recall | 0.416 | 0.466 | 0.244 | 0.332 |
| F-measure | 0.381 | 0.343 | 0.248 | 0.209 |

### 4.3. Tests

We performed a series of tests for Concept Hierarchy Refinement using the general schema described earlier. Two sampling strategies were tested:

1. Random selection. Three different objects are randomly picked from the set of all objects, with uniform distribution. This is not an active strategy, since we do not use information from the data set. It is equivalent to providing all the labeled samples to the algorithm at the beginning.

2. An ensemble strategy combining dense samples with random samples. Specifically, 50% of triples where chosen by first selecting a random concept from the hierarchy, then taking random sample from within that concept. The other 50% where chosen by random selection from the entire set, like in the first strategy. This ratio was determined by preliminary testing. Choice between the two methods is made randomly during each execution of `ActiveSelect`. Additionally, previously queried samples were periodically used to refine the hierarchy with known answers. Those operations did not require any involvement from an expert.

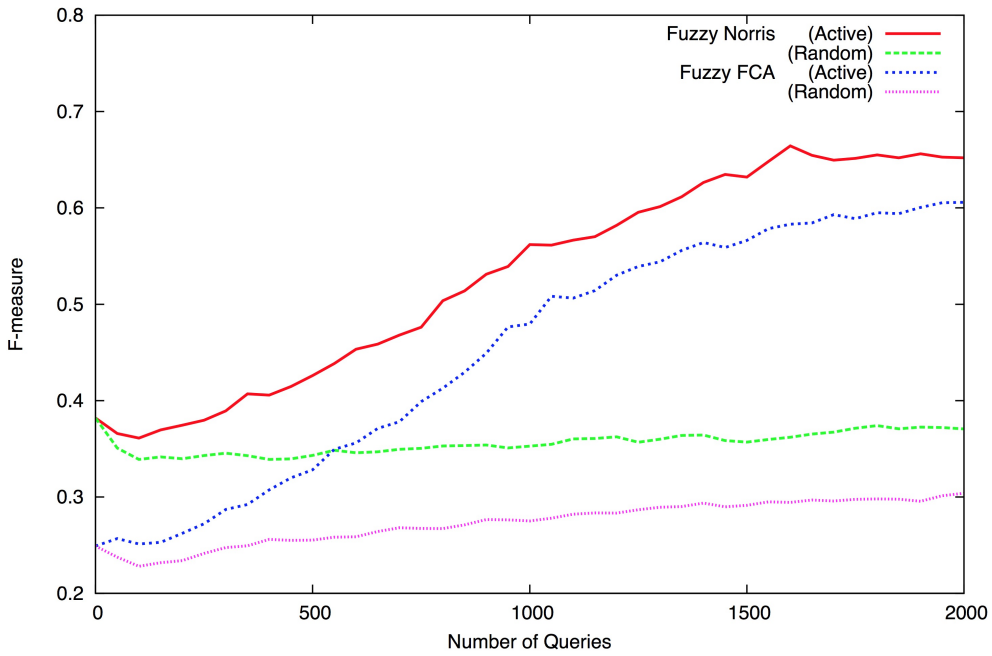
For both unsupervised algorithms, parameters where chosen to get hierarchies of a desired size. Fuzzy Norris' algorithm was executed for *tolerance* equal 0.55 and for 0.6. In the first case, 33 concepts where generated, in the second one – 48 concepts. Fuzzy FCA was used with cut-off threshold at 0.02. Clustering threshold was set to 0.35 and 0.36, to get hierarchies of 27 concepts and 61 concepts, respectively. The 4 initial hierarchies were measured against our manually generated hierarchy. The results are presented in Table 1. More thorough tests of both algorithms for a wide range of parameters where performed in (Dereziński, 2012). In this paper, we chose the values that are representative of the overall performance of each algorithm.

Each test consisted of 2000 queries and was repeated 10 times (for different random selections). During each of the 10 runs, Precision, Recall and F-measure, as well as hierarchy size, were calculated every 50 queries. To make performance charts with respect to number of iterations, averages of each measure were taken accross 10 runs.

Table 2: Average F-measure per number of queries for each test case.

| | Fuzzy Norris | | | | Fuzzy FCA | | | |
|---|---|---|---|---|---|---|---|---|
| | Active | | Random | | Active | | Random | |
| | 33 | 48 | 33 | 48 | 27 | 61 | 27 | 61 |
| 0 | 0.381 | 0.343 | 0.381 | 0.343 | 0.248 | 0.209 | 0.248 | 0.209 |
| 500 | 0.426 | 0.424 | 0.343 | 0.320 | 0.328 | 0.306 | 0.255 | 0.228 |
| 1000 | 0.561 | 0.523 | 0.352 | 0.329 | 0.479 | 0.412 | 0.275 | 0.248 |
| 1500 | 0.631 | 0.561 | 0.356 | 0.339 | 0.566 | 0.501 | 0.291 | 0.254 |
| 2000 | 0.651 | 0.586 | 0.370 | 0.351 | 0.605 | 0.519 | 0.303 | 0.262 |

Figure 3: F-measure active learning comparison for Fuzzy Norris hierarchy with tolerance 0.55 and Fuzzy FCA with clustering threshold 0.35.




### 4.4. Results

The results of the tests for all configurations can be observed in Table 2. We can see significant improvement of F-measure across all the tests with active learning, even after just 500 queries. In many tests there is still significant upward tendency after 2000 queries. For the initial hierarchy generated by Fuzzy Norris algorithm with *tolerance* 0.60, the average improvement after 500 queries with active learning is 0.08. After 2000 queries, F-measure increased by over 0.24 to reach 0.59 for the active approach, and by 0.01 with random sampling. So, we can clearly see that our active strategy has a significant positive influence on the results. For the hierarchy with *tolerance* 0.55, the benefit of active learning is even more apparent. After 500 queries, the result is almost 0.05 higher than for the initial hierarchy, while after 2000, the improvement is over 0.27, with F-measure reaching 0.65.

As we can see, the results with initial hierarchy generated by the Fuzzy Formal Concept Analysis approach were also very good, despite that algorithm not performing very well by itself. Average F-measure increased from 0.25 to 0.61 after 2000 queries, when using active learning on the hierarchy with clustering threshold 0.35. Figure 3 shows direct comparison of F-measure results for active strategy against random sampling for both Fuzzy Norris and Fuzzy FCA.

Interestingly, the results show that during the first 100 iterations, the Hierarchy Refinement algorithm in some cases causes F-measure to drop slightly, before it starts increasing. To examine this behaviour, Figure 4 shows more detailed plots with Precision, Recall and F-measure for a test which displayed such behavior. As it turns out, initially the procedure

Figure 4: Precision, Recall and F-measure results for Fuzzy Norris hierarchy with tolerance 0.55 and active sampling.

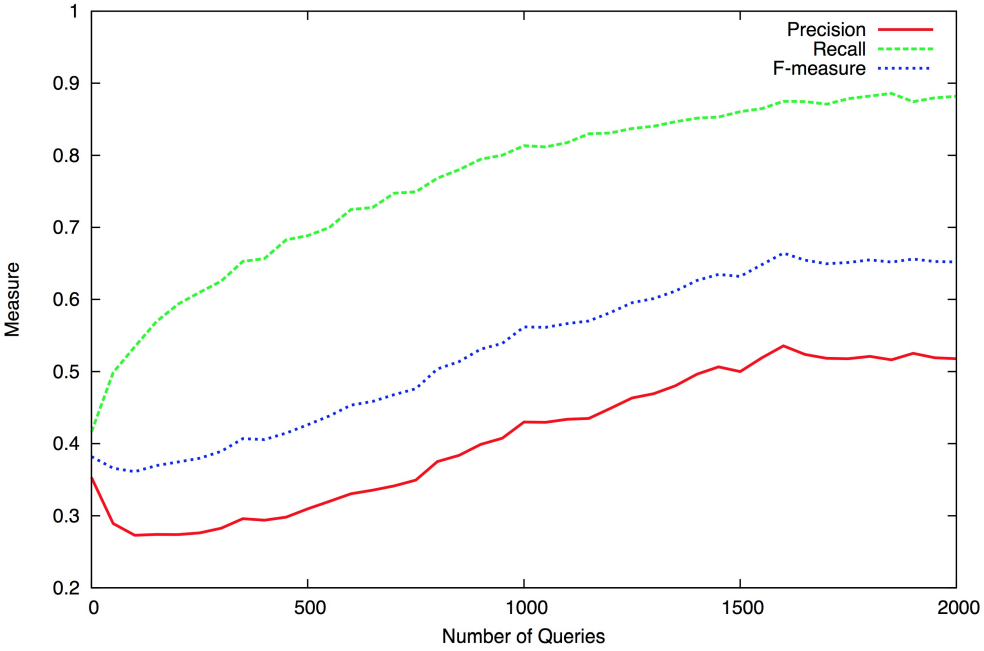
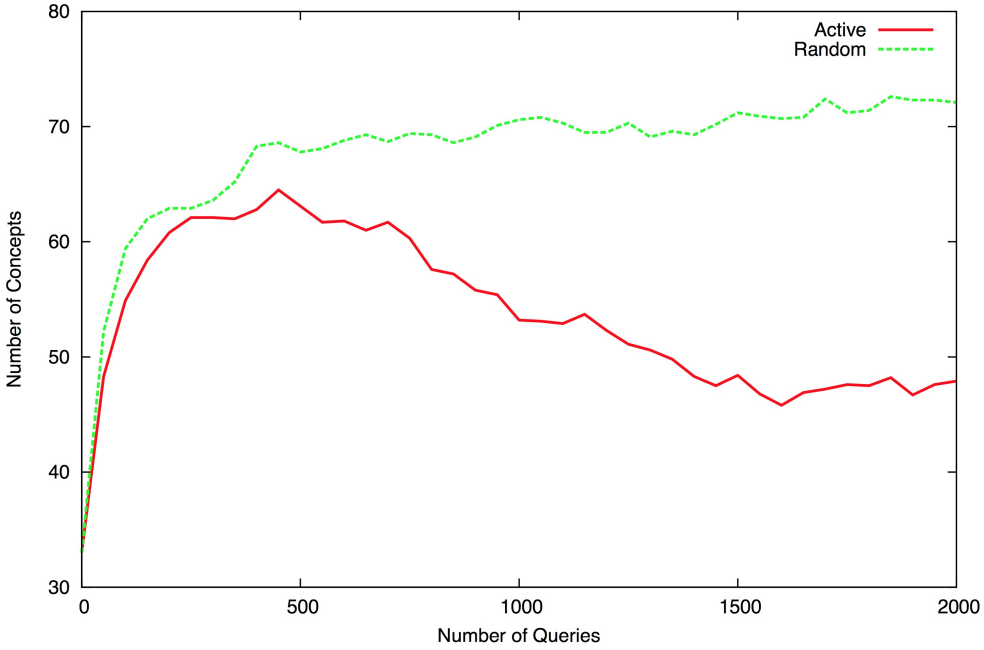



Figure 5: Number of concepts per number of queries for Fuzzy Norris hierarchy with tolerance 0.55 for both active and random sampling.



improves Recall significantly, while sacrificing Precision. Then, Precision starts to increase as well, while Recall continues to improve, albeit more slowly. This suggests that hierarchy refinement initially creates more new concepts than are removed. Indeed, that is what happens, as shown on Figure 5. In all of the tests with active learning, the hierarchy size initially increased and after reaching certain level, began falling much more slowly. It seems that the process first needs to achieve certain level of concept saturation within the hierarchy, before stabilizing. Notice, that our active learning strategy strongly facilitates hierarchy size reduction during that second phase, compared to random sampling. For each query, whether the refinement will add or remove a concept depends partially on the predicted answer. If `Compare` is able to choose object which does not fit with the other, than after examining all the possible outcomes we know for certain that hierarchy size will not increase and it may in fact decrease. However, if the function returns None, then depending on the expert's answer, the hierarchy size may increase or stay the same. As the hierarchy size increases during the early queries, likelihood of the prediction being None becomes smaller, which is why at some point that trend changes.

## 5. Conclusion

Concept hierarchy generation is a very relevant problem with many applications. In this paper, we have proposed a technique for active semi-supervised concept hierarchy refinement which can be used in conjunction with any unsupervised algorithm. We have described a system for quering human experts, which requires much less cognitive effort from the experts than other existing solutions, and therefore allows for more queries to be processed in a given amount of time. Furthermore, an algorithm was developed which can adjust given hierarchy to better match the result of each query. The proposed method is very suitable for using active learning to improve the results and minimize human effort by intelligently choosing which queries should be presented to the expert. Several possible active learning techniques are discussed with regard to this model.

The method was evaluated for the task of creating a concept hierarchy from document topics. For generating the initial concept hierarchy, we used two unsupervised algorithms: Fuzzy Formal Concept Analysis and Fuzzy Norris algorithm. To compare the results, a hierarchical similarity measure was used in conjunction with a manually created topic hierarchy, which was also used to simulate human responses to the queries. The concept hierarchy refinement algorithm was tested with an active learning strategy, which uses density sampling to provide better queries for the expert. The strategy uses only the structure of generated hierarchy to generate samples, so it can be applied to any hierarchy generation task. It also means that the strategy should improve overtime along with the concept hierarchy.

The results were presented for concept hierarchies refined with up to 2000 queries. The method proved to be very effective at improving both Recall and Precision of generated concepts. The charts show that proposed active learning strategy is a key element for this approach to reach satisfactory performance. The results also show how universally applicable the method is, performing equally well with both unsupervised algorithms, as well as for various initial hierarchy sizes. The technique provided valueable outcome even for a relatively small amount of samples, considering that for many tasks one such query can be answered in just a few seconds, with a proper user interface. After 500 queries, F-measure

increased on average by 40% for Fuzzy FCA algorithm and about 18% for Fuzzy Norris. For 2000 queries the average improvement was around 70% for Fuzzy Norris and 146% for Fuzzy FCA. The model proposed in this paper is highly modular and extendable. There is a lot of potential for future research in this area to further improve its performance. Firstly, although the technique works in such a broad range of problems, focusing on one specific task should enable making more adjustments to the algorithm, based on the information given for each data object. For example, many aspects of this method depend on a total relation between concepts, that needs to be defined on top of the natural subsethood relation. Choice of this method is highly dependent on the initial data distribution. Finally, much more sophisticated active learning strategies could be devised if we combined the initial information about each data point with the structure of the refined concept hierarchy.

## References

Avrim Blum, Tom Mitchell.
Combining labeled and unlabeled data with co-training.
*Proceedings of the eleventh annual conference on Computational learning theory*, 1998.

Bernhard Ganter, Rudolf Wille.
*Formal concept analysis. Mathematical Foundations.*
Springer-Verlag, Berlin, 1999.

Guan T. Tho, Siu C. Hui, A. C. M. Fong, Tru H. Cao.
Automatic Fuzzy Ontology Generation for Semantic Web.
*IEEE Transactions on Knowledge and Data Engineering*, 18(6), 2006.

Ruggero G. Pensa, Jean-François Boulicaut.
Towards Fault-Tolerant Formal Concept Analysis.
*AI*IA Advances in Artificial Intelligence*, LNAI, 3673, p. 212–223, 2005.

Michał Dereziński.
*On Generating Concept Hierarchies with Fuzzy Data.*
http://students.mimuw.edu.pl/~md262929/cs-masters.pdf
Master's thesis, University of Warsaw, Poland, 2012.

Sugato Basu, Arindam Banerjee, Raymond Mooney.
Semi-supervised Clutering by Seeding.
*Proceedings of the 19th International Conference on Machine Learning*, pp. 19-26, 2002.

Nizar Grira, Michel Crucianu, Nozha Boujemaa.
Unsupervised and Semi-supervised Clustering: a Brief Survey.
*A Review of Machine Learning Techniques for Processing Multimedia Content*, 2005.

Korinna Bade, Andreas Nürnberger. Creating a Cluster Hierarchy under Constraints of a Partially Known Hierarchy.
*Proceedings of the 2008 SIAM International Conference on Data Mining*, 2008.

Xian-Ling Mao, Zhao-Yan Ming, Tat-Seng Chua, Si Li, Hongfei Yan, Xiaoming Li.
SSHLDA: A Semi-Supervised Hierarchical Topic Model.
*Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 800-809, 2012.

Bruno M. Nogueira, Alipio Jorge, Solange O. Rezende.
A confidencce-based active approach for semi-supervised hierarchical clustering.
*ICMC Technical Reports*, 2011.

Pinar Donmez, Jaime G. Carbonell.
Proactive Learning: Cost-Sensitive Active Learning with Multiple Imperfect Oracles.
*ACM 17th Conference on Information and Knowledge Management*, 2008.

Kacper Chwiałkowski.
*Ekstrakcja Tematów.*
http://students.mimuw.edu.pl/~kc262414/pracamgr.pdf
Master's thesis, University of Warsaw, Poland, 2012.

E. M. Norris.
An algorithm for computing the maximal rectangles in a binary relation.
*Revue Roumaine Mathématiques Pures et Appliquées*, 23(2), pp. 243–250, 1978.