Learning Implicit References to Math Concepts in Computing Literature

Deshpande, Amol **AMOLD1@UMBC.EDU**

Peshave, Akshay **PESHAVE1@UMBC.EDU**

University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21227 USA

1. Introduction

Applied sciences emerged as the need to create practical solutions to real-world problems was felt as much as the need to advance the theory of pure sciences. Applied sciences are varied amalgamations of pure sciences, computer science and nanotechnology being classic examples. Analogous is the new-age notion of interdisciplinary studies. Consequently, understanding of and research in these applied fields requires transcending boundaries into the theory of one or more closely related classical sciences. Progressively, frequently occurring references to concepts of an underlying pure science for established applied science notions are rendered implicit. There is a need to identify and explicitly suggest such implicit pure science concepts to learners and naïve readers of the applied sciences' literature.

With this in view we are attempting to develop a Proof-of-Concept model to learn implicit underlying references to mathematical concepts for a variety of computing science notions.

2. Related Works

2.1 Document Classification and Categorization

Document classification is the assignment of predefined categories to documents based on their content and/or context. Automation of this task can be supervised, semi-supervised or un-supervised.

2.2 Semantic Integration and Ontology alignment

This area deals with interrelating information between heterogeneous data sources. Heterogeneity may be exhibited in structure, context or value. Metadata is often used as an enabler to derive links based on semantic similarity and appropriate rules with relatively high confidence. This is a semantic technology dependent approach which deals with construction of association graphs and their traversal for resolving inferences.

2.3 ITM Align

This is a Mondeca product which augments their Intelligent Topic Manager (ITM) to align reference vocabularies, such as various language thesauri, to achieve data integration. This solution is again based on semantic technologies and is a good example of semantic matching and alignment.

3. Proposed Methods

The crux of our problem is semi-supervised learning of a prediction model for semantic associations between computing science and mathematical concepts. Intuitively, we should look for explicit hints of any such associations and reinforce our beliefs with increasing occurrences of these explicit hints. A rich source of these explicit references are the metadata associated with research publications in the field of computing science. We have chosen:

1. American Computing Society (ACM) as a source for computing science research artifacts.
2. American Mathematical Society (AMS) as a references for any mathematical terms and concepts we may need.

The solution can be modeled in 3 distinct steps.

3.1 Retrieving Explicit CS → MT Concept Associations

Strong explicit references to MT concepts exist in ACM publications in the following forms:

1. User Defined Tags (UDT) which may contain MT concepts
2. Explicitly mentioned MT concepts (MTC) in the abstract of the publication

These references can be resolved by a look-up of UTs and Abstract Terms (AT) and in the Mathematics Subject Classification (MSC) published by AMS. The retrieval of required data proceeds is highlighted below:

1. Extract ACM Computing Classification System (CCS) tags, UDT and Abstract from the PDF versions of the publications.

2. Extract AT from the abstract using NLP-based Alchemy API.
3. Look-up UTs and ATs in MSC to produce explicit MTCs.

3.2 Creating A Learning Data Set Of Explicit Association Rules

The above phase yields the sets CCS, UT, AT and associated MTC. We can cleanse these by applying the below operations:

UCT = UDT – user defined tags present in MSC

ACT = AT – UCT – abstract terms present in MSC

We further restrict the length of the sets by choosing the top priority items based on measures such as relevance scores and frequency of occurrence. We are then left with the following sets:

$CCS = \{ccs_1, ccs_2, ccs_3, \dots , ccs_i\}$

$UCT = \{uct_1, uct_2, uct_3, \dots , uct_m\}$

$ACT = \{act_1, act_2, act_3, \dots , act_m\}$

$MTC = \{mtc_1, mtc_2, mtc_3, \dots , mtc_j\}$

These sets allow us to construct rules of the form:

$UCT \cup ACT \rightarrow ccs_i$ (text classification)

$\underline{UCT \cup ACT \rightarrow mtc_j}$ (text classification)

$ccs_i \rightarrow mtc_j$ (domain integration)

The data-set consisting of rules from the previous phase can be learned to yield classification model for each of these rule forms.

3.3 Learning a Classification Model

Input to the classifier, with or without explicit mention of math terms, could be:

1. Textual content, in which case we need to extract terms from the content using Alchemy API
2. A list of terms to be used directly by the predictor.

The key aspects that need to be addressed are:

1. In machine learning algorithms the feature set is a ordered vector. But for the current problem the feature values are ideally unordered tuples. This may lead to a loss of generality.
2. Each input text or list of terms may not be of the same length as the training feature set. We need to be able to predict MTCs with some, maybe relatively lower, probability.
3. We should be able to suggest multiple MT concepts with a relevance score based on the input.

Possible classifying models that may be best fits given above aspects are:

1. Bag of words approach (in the favor of simplicity)
2. For each tuple in the rule-set introduce generate an extra rule for each permutation of the feature values. This would enable usage of other learning models such as Support Vector Machines. This approach though may lead to over-generalization which would adversely affect the accuracy of the method.
3. A customized fusion of k-Nearest Neighbor and decision trees on the rules data-set as generated in 3.2

4. Experiment Design

4.1 The Data Set

Choice of ACM papers is crucial to validating how accurately the proposed model works. We begin with a training set of 50+ most cited and downloaded research papers in select ACM forums. These select forums are ones which have a high dependency on mathematical theories for their advancement, such as ACM Transactions on Algorithms (TALG) , ACM Transactions on Knowledge Discovery from Data (TKDD), ACM Transactions on Computation Theory (TOCT) etc.

Manually downloaded citation pages from ACM Digital Library are run through our HTML parsing module. The module extracts required meta-data about the publications which includes user keywords and the document itself.

The user keywords are processed using Alchemy API, an online NLP based entity extraction framework. This removes noise words/phrases, concept-based redundant keywords and keywords added for the purpose of SEO. The output of this phase is the union of UCT and ACT.

The user keywords are also looked up in the MSC specification using a full-text search. The 5 most relevant search results are recorded as MTC.

The document is processed via a PDF parser to extract text and further find the CCS classification for the publication.

4.2 The Training Set

The union of UCT and ACT is modeled as a bag-of-words. The bag size for our sample data set of 54 documents is 259 phrases.

Although the RHS of all rules discussed in 3.2 are vectors themselves, we haven't modeled them as a bag-of-words. Instead we have introduced one training tuple per element in the RHS vector. This simplifies things at the expense of precision.

The training set has been created in ARFF and SVM formats for experimenting using various learning models in Weka and SVM^{multiclass}.

Below are tables stating some facts about the data set to help understand it's representativeness of the problem discussed. It also helps highlight the vastness and complexity of this learning problem and the relative sparseness of the data-set used for this PoC.

Publication Count	54
Feature-set size i.e. Bag-of-words size	259

Table 1: Data-set Specifics

Tags → MSC mapping	270
Tags → CCS mapping	113

Table 2: Training Set Sizes

Analysis of Algorithms and Problem Complexity
Information Storage and Retrieval
Document and Text Processing
Pattern Recognition
Database Management
Artificial Intelligence
Nonnumerical Algorithms and Problems

Table 3: Representative CS Knowledge Areas of Data-set

4.3 The Validation Set

The validation set is created by introducing random noise in samples from the training set keeping majority of the bag-of-words flags the same and validate correct predictions.

4.4 Choosing The Learning And Classifying Model

We have experimented with several classification models to learn this data-set. The key challenges for the selection process in this PoC have been:

1. Difficulty in restricting the scope of the PoC to a few CCS knowledge areas and even fewer MSC knowledge areas therein.
2. Large feature-set in spite of the knowledge area restrictions.
3. Sparseness of the feature-set values.
4. Low representativeness of the true mappings by the limited training instances. This is due to the lack of availability of the publications themselves via an automation framework or set of APIs.

Probabilistic Theory
Metric Theory Of Algorithms
Enumerative Combinatorics
Probabilistic Methods In Group Theory
Graph Theory
Theory Of Data
Computability And Recursion Theory
Complexity And Performance Of Numerical Alg.
Proof Theory And Constructive Mathematics

Table 4: Representative Math Knowledge Areas of Data-set

We chose to experiment with training on two data-sets:

1. Singleton data-set : Mapping of each document to a single class label.
2. True data-set: Mapping of each document to multiple class labels i.e a vector of class labels represented as multiple training instances.

The first approach results in a more confident classifier model which fits the training set well. The second approach reduces the confidence levels substantially and is unable to fit the training set itself. This is due to the presence of training instances mapping the exact same feature values to multiple class labels.

We used two training models which performed relatively well on the data-set, Grid Search and multi-class SVM.

4.5 Grid Search

We use Grid Search with linear regression for our base experiments. Grid Search on the singleton data-set fits the training set accurately with 0.0019% root relative squared error. Validation set is classified by this model with approximately 30% root relative squared error. This is an indication of the low representativeness of the training data-set.

Grid Search on the true data-set under-fits severely, as expected. Root relative squared error in this case is 81%. Bagging with Grid Search reduces root relative squared error marginally to 79% and relative absolute error by 4%.

4.6 SVM^{multiclass}

We applied multi-class Support Vector Machine as our second classifier model. We used the SVM^{multiclass} open-source implementation for these experiments. All experiments were conducted with 1.0 training error and margin tradeoff.

We trained the SVM model on singleton as well as true data-sets using linear, quadratic and RBF kernels. Below are tables detailing the empirical results in each case.

	#SV	Margin	Test-set Accuracy
Linear	4	2.52	100.00%
Quadratic	1	41.96	100.00%
RBF (γ=0.5)	1	497.51	100.00%

Table 5: Singleton Data-set Results

	#SV	Margin	Test-set Accuracy
Linear	3	4.66	10.00%
Quadratic	5	9.74	20.00%
RBF (γ=0.5)	1	6.27	50.00%

Table 6: True Data-set Results

The results for singleton results are evidently misleading. A validation instance with same feature values but another one of the possible MSC classifiers will yield 0% accuracy.

The results for true data-set modeling reflect the low confidence and confusion of the modeled classifier due to the presence of multiple instances with exactly same feature values but different class labels.

The ideal scenario would be a model which would learn the true data-set in accordance with the probabilities or relevance of each class in the class label vector. The model should then be able to predict the class label with the approximate confidence/relevance of each label in the class-label vector.

Alternatively, we can cluster all the training samples based on their feature values and then predict a cluster to knowledge area mapping. This again is possible when we have a dense data set representative of true mappings between keywords and class labels. The current training set is not well distributed based on the results from a Expectation Maximization clustering agent. The results are provided in Appendix A. The clustering has been done based on means and relative standard deviation of the feature values for instances in each cluster.

5. Conclusion

We have seen that although this model is effective on the singleton data-set it is not effective on the redundant instance approach to express the true data-set. The highest achievable accuracy for our restricted data corpus using the redundant instance approach is 50% using the RBF kernel SVM.

Additional metadata regarding the relevance/weight of each CCS and MSC classes the publications are mapped to, if made available by the ACM Digital Library, can help implement and evaluate the approaches suggested at the end of 4.6.

6. Future Work

Higher accuracy can be achieved by treating the class labels as vectors rather than duplicating feature values for each class label in the training set. A neural network can be trained using this true data-set to achieve a better classifier.

One of the key features that could be implemented is associating a real value with each item in the class-label vector indicating their relevance or strength in the classification taxonomy hierarchy for a given publication.

Further merging of classifiers for CCS and MSC to obtain a CCS $\rightarrow$ MSC mapping will be very useful to augment the system.

References

Sebastiani, F. *Machine Learning in Automated Text Categorization.,* 2001, [ONLINE] Available `http://arxiv.org/pdf/cs.ir/0110053.pdf`

Doan, AnHai, Jayant Madhavan, Pedro Domingos, and Alon Halevy. "Ontology matching: A machine learning approach." *Handbook on ontologies in information systems* (2004): 397-416 [ONLINE] Available `http://homes.cs.washington.edu/~pedrod/papers/hois.pdf`

Modeca Inc. , *Architecture and System Requirements of ITM,* [ONLINE] Available `http://www.mondeca.com/content/download/319/10565/file/Architecture%20and%20system%20requirements.pdf`