Adaptive Organization of Digital Documents using Knowledge Graphs

submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

of the
Indian Institute of Technology Bombay, India
and
Monash University, Australia
by

Ramakrishna B Bairi

Supervisors:

Dr. Ganesh Ramakrishnan (IIT Bombay)
Dr. Mark Carman (Monash University)

The course of study for this award was developed jointly by the Indian Institute of Technology Bombay, India and Monash University, Australia and given academic recognition by each of them. The programme was administered by The IITB-Monash Research Academy.

2017

to my Parents, Wife and Children

Approval Sheet

The thesis entitled "Adaptive Organization of Digital Documents using Knowledge Graphs" by Ramakrishna B Bairi is approved for the degree of **Doctor of Philosophy**

(CHIRANJEEB · B.)
External Examiner

(SOUMEN)
Internal Examiner

(GANESH · R.)
IITB Supervisor

(MARK CARMAN)
Monash Supervisor

(MALHAR KULKARNI)
Chairman

Date: 28/07/2017
Place: MUMBAI

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute/the Academy and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Notice 1

Under the Copyright Act 1968, this thesis must be used only under normal conditions of scholarly fair dealing. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.

Notice 2

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owners permission.

IITB ID: 114054001
Monash ID: 23759291

Mumbai	
Place	Signature
28-July-2017	**Ramakrishna B Bairi**
Date	Name

Abstract

In this thesis we study the problem of evolving a hierarchy of categories that best organize the documents in a collection under soft (preferential) and hard (necessary) constraints, using a massive knowledge graph. The evolved categorization should be neither under-specified nor over-specified and capable of adapting to the user preferences and temporal changes in the knowledge graph. Under-specified categories can result when an insufficient number of categories is used to describe the documents in the collection. Meanwhile, over-specified categories result when too many categories are used to describe the documents in the collection. Under-specified organization usually arises when adopting an existing set of categories and forcing the document collection to fit under them. For example, the organization the of Reuters news articles [84] into the category structure provided by 20NewsGroup [1] results in an under-specified organization. Over-specified organization can result when every important keyphrase in a document is promoted as a category for that document. For example, making use of all Hashtags from Twitter could result in too many keyphrases to describe the document collection thus provide an over-specified organization. Apart from addressing these two problems, this thesis also presents a few ideas for evolving a categorization that adapts to the user interests and changes to the knowledge graph over time.

Using the concepts from the knowledge graph as candidate categories, we model the problem of category identification for a document as that of inferring binary labels in an Associative Markov Networks (AMN)[139]. The AMN based framework allows us to encode global and local features, including user preferences, and allows us to optimally assign categories for each document in the collection. To learn the user preferences and to improve the system accuracy, we propose an Active Learning model in the AMN framework. In Active Learning, we present many ideas for identifying uncertain categories and

documents, and discuss a framework for joint Active Learning in the category-document space. While category identification for each document serves as a high recall step, it suffers from the over-specified organization problem. This is then addressed through our category summarization framework using Submodular Mixtures. By defining various submodular coverage, diversity and quality control barrier functions over the knowledge graph structure, we present a technique for learning to summarize the over-specified categories into a smaller set of categories. While the above technique is based on supervised learning, in this thesis we also present ideas for unsupervised techniques using Sampling and Expectation Maximization, for identifying a smaller DAG structured categories for organizing the document collection

Since short texts (for example, tweets, reviews, and the like) are becoming abundant nowadays, organizing them effectively is important to many downstream applications such as retrieval, ranking, search, etc. However, short texts are different from long text documents due to very limited context they contain. This makes organization of short texts challenging. In this thesis we present a novel technique for expanding them in such a way that the expansion improves the accuracy of the downstream application, including short text organization.

Publications

1. "A Framework for Task-specific Short Document Expansion", Ramakrishna Bairi, Raghavendra Udupa, Ganesh Ramakrishnan, In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, Indianapolis, USA, Oct 24-28, 2016.

2. "Beyond clustering: Sub-DAG Discovery for Categorising Documents", Ramakrishna Bairi, Mark Carman, Ganesh Ramakrishnan, In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, Indianapolis, USA, Oct 24-28, 2016.

3. "Multi-Topic Summarization in DAG-Structured Topic Hierarchies via Submodular Mixtures", Ramakrishna Bairi, Ganesh Ramakrishnan, Rishabh Iyer, and Jeff Bilmes. In Proceedings of the Association for Computational Linguistics/Asian Federation of Natural Language Processing (ACL-IJCNLP), Beijing, China, 2015.

4. "On the Evolution of Wikipedia: Dynamics of Categories and Articles", BAIRI, R.; CARMAN, M.; RAMAKRISHNAN, G., International AAAI Conference on Web and Social Media, North America, Apr. 2015.

5. "Categorising videos using a personalised category catalogue", Ramakrishna B. Bairi, Ankit Vani, Pooja Ahuja, and Ganesh Ramakrishnan. 2015. Categorising videos using a personalised category catalogue. In Proceedings of the Second ACM IKDD Conference on Data Sciences (CoDS '15). ACM, New York, NY, USA, 49-58.

6. "Personalized Classifiers: Evolving a Classifier from a Large Reference Knowledge Graph", Ramakrishna B Bairi, Ganesh Ramakrishnan, Vikas Sindhwani, IDEAS'14 July 07-09 2014, Porrto, Portugal.

7. “Labeling Documents in Search Collection: Evolving Classifiers on a Semantically Relevant Label Space”, Ramakrishna B Bairi and Ganesh Ramakrishnan, SIGIR 2014 Workshop on Semantic Matching in Information Retrieval, July11, 2014, Queensland, Australia.

8. “Learning to Generate Diversified Query Interpretations using Biconvex Optimization”, Ramakrishna Bairi, Ambha A, Ganesh Ramakrishnan. , Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 733-739, Nagoya, Japan, October 2013, Asian Federation of Natural Language Processing.

9. “Rel-Div: Generating Diversified Query Interpretations from Semantic Relations”, Ramakrishna Bairi, Ambha A, Ganesh Ramakrishnan, 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013

10. “Effective Mentor Suggestion System for Online Collaboration Platform”, Advait Raut, Upasana Gaikwad, Ramakrishna Bairi and Ganesh Ramakrishnan, Proceedings of the Workshop on Speech and Language Processing Tools in Education, December 2012, pages 1-8, COLING 2012.

11. “SATTY : Word Sense Induction Application in Web Search Clustering”, Satyabrata Behera, Upasana Gaikwad, Ramakrishna Bairi, Ganesh Ramakrishnan, SemEval 2013

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As digital data grows in the form of news, blogs, web pages, scientific articles, books, images, sound, video, tweets, quotes, and so on – which we collectively term "digital documents" – the need for effective organization of the documents into hierarchical categories becomes self-evident and imperative. Such an organization will be useful in a number of downstream applications in Information Retrieval (IR), Natural Language Processing (NLP) and Machine Learning (ML), such as diversification of search results, exploratory search, entity disambiguation, tag cloud generation and e-Catalog creation.

1.1 Challenges in Evolving Categorization

For effective organization, it is very important to create a right set of categories that represent the contents in the collection. Imagine a collection with thousands of articles on various topics. In order to set up a good categorization system, we need to first develop a set of categories that can effectively represent the documents in the collection. Categories that are generic, such as News, Entertainment, Technical, Politics, Sports, and such others might not be of much use. Each of such categories accumulates thousands of articles and searching for the required piece of information continues to be challenging. On the other hand, fine grained category creation needs domain experts and is a laborious task. Hence identifying the right set of categories becomes the foremost challenge.

1.1.1 Under-specified Organization

One option for category selection is to make use of predefined category systems. For example, categories from Reuters RCV1 Text Collection [84], Yahoo Directories[1] or DMOZ [47] can be adopted and the existing documents in the collection can be assigned into these categories. We conducted an experiment by choosing predefined categories from the 20NewsGroup [1] dataset. We learned a classifier using Support vector machines (SVM) classifier by using the 20NewsGroup training set and then used it to classify the Reuters-21578 [115] documents. The Table 1.1 shows the distribution of document over these categories. The resulting distribution of the documents over the categories is highly skewed. Some of the categories are assigned very few articles and some too many articles. This happens because a very small number of categories is relevant to the document collection. We term this scenario an under-specified organization, in which the number of categories are too small to effectively cover the documents in the collection. This indicates that adoption of predefined categories may not work in many situations

1.1.2 Over-specified Organization

Predefined categories can pose another challenge – over–specified organization. When the number of predefined categories are too many for the document collection, each document is assigned in its own category which lead to very sparse and less interpretable categorization. For example, if we take several thousand predefined Wikipedia[2] categories and classify a few thousand documents from the 20NewsGroup collection using them, many of the categories are assigned to very few or no documents. We conducted an experiment by running WikipediaMiner[3] system on 20NewsGroup document collection and inspected the categories assigned. WikipediaMiner assigned approximately 9000 categories, of which 40% had less than two documents, 80% had less than 5 documents and only 2% had more than 10 documents categorized under them. The skew and sparsity of the document distribution across the categories happens due to very large number of categories are

[1] `https://en.wikipedia.org/wiki/Yahoo!_Directory`

[2] `https://en.wikipedia.org`

[3] `http://wikipedia-miner.cms.waikato.ac.nz/`

20NewsGroup Categories	% RCV1-v2 documents	20NewsGroup Categories	% RCV1-v2 documents
alt.atheism	0	rec.sport.hockey	2.5
comp.graphics	4.5	sci.crypt	1
comp.os.ms-windows.misc	0	sci.electronics	4.4
comp.sys.ibm.pc.hardware	4	sci.med	3.5
comp.sys.mac.hardware	2.3	sci.space	8.7
comp.windows.x	0	soc.religion.christian	4.8
misc.forsale	22	talk.politics.guns	1
rec.autos	10.2	talk.politics.mideast	1
rec.motorcycles	0	talk.politics.misc	16.8
rec.sport.baseball	13.3	talk.religion.misc	0

Table 1.1: Distribution of Reuters RCV1-v2 documents on 20NewsGroup categories

used to describe the document collection and such a categorization may not be useful for downstream applications.

1.1.3 Intent Coverage

Users of a categorization system may want to include personal preferences while creating the categories. That is, one may want less fine grained categories in some areas and more fine grained categories in other areas. For example, an institute from the field of computer science may not wish to categorize publications down to the level of individual bacteria or genes, even though there are documents in their collection discussing classification algorithms for bacteria or genes. The institute may prefer to categorize all its documents into computer science sub-fields. The opposite could be true for an institute from the field of bio technology.

1.1.4 Temporal Relevance

Another practical challenge in a predefined or fixed category system is the maintaining of the temporal relevance of the category system. The categories that can accommodate the documents in the collection today may need more categories tomorrow when new documents are added to the collection. A good categorization system should detect the emergence of these new categories and evolve the classifier to accommodate the new categories. For example, a digital library at present may not have any documents regarding "Computer Vision". However, as time progresses, it may accumulate a large number of documents on "Computer Vision". A good document organization system should detect this and organize those documents under a new category "Computer Vision".

This thesis describes techniques that can enable the overcoming of the challenges mentioned above. Before describing our approach, we review the literature on document categorization techniques

1.2 The Literature on Document Organization

To the best of our knowledge, the holistic approach in building an organization specific digital document organization procedure introduced in this thesis is new. Previous work on document organization can be broadly identified under the following areas: Document Classification, Knowledge Mining and Topic Modeling. We now discuss the literature on each of these area.

1.2.1 Document Classification

Document classification is the act of dividing a set of input documents into two or more classes in which each document can be said to belong to one or multiple classes. These classes are generally predefined. ML techniques [164] are employed to automatically build a classifier by learning from a set of pre-classified documents that represent the characteristics of the classes. This approach has an advantage over the knowledge engineering

approach of the manual definition of a classifier by domain experts, due to considerable savings in terms of expert labor power and straightforward portability to different domains.

When categorizing a document, a computer program often treats the document as a "bag of words". It does not attempt to process the actual information as information extraction does. Rather, categorization only counts words that appear and, from the counts, characterizes the main topics that the document covers. By using supervised learning algorithms [31], the objective is to learn classifiers from known examples (labeled documents) and to perform the classification automatically on unknown examples (unlabeled documents). During the training phase, a set of labeled documents are prepared, in which each document is manually assigned one or more classes from a set of predefined classes. In the training process, the ML algorithms build a model and learn its parameters which will maximize the likelihood of the training data. Next, by using the learned models, new documents are classified. During this inference stage, new documents are assigned labels from the predefined set of labels. However, this time the computer program assigns the labels without the need for manual intervention.

Various document categorization techniques have been proposed in the literature for supervised classification tasks. Some of the widely used techniques are discussed briefly below.

1.2.1.1 Decision Trees

Decision tree methods rebuild the manual categorization of the training documents by constructing well-defined true/false queries. The queries are in the form of a tree structure in which the nodes represent questions and the leaves represent the corresponding category of documents. After the tree is created, a new document can easily be categorized by putting it in the root node of the tree and letting it run through the query structure until it reaches a certain leaf. Notable decision tree algorithms are ID3 [111], C4.5 [112], CART [23], CHAID [72], MARS [57] and Conditional Inference Trees [65]

1.2.1.2 k-Nearest Neighbors

In this approach, the categorization of a document is performed by comparing the category frequencies of the k - nearest documents (neighbors)[39]. The majority category from the k nearest documents is assigned as the category of the document that is being categorized.

The training examples are vectors in a multidimensional feature space, each with a predefined class label. The training phase of the algorithm consists only of the storing of the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label that is most frequent among the k training samples that are nearest to that query point.

1.2.1.3 Bayesian Methods

One of the early document classification techniques is Naive Bayes. The naive part of the Naive Bayes classifier is the assumption of word (that is feature) independence, which means that the word order is irrelevant and consequently that the presence of one word does not affect the presence or absence of another word. This still remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, and the like.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including SVM [135].

1.2.1.4 Neural Networks

Neural networks consist of many individual processing units termed as neurons, which are connected by links that have weights that allow neurons to activate other neurons. Different neural network approaches have been applied to document categorization problems [35, 80, 93]. While some of them use the simplest form of neural networks, (known as perception), which consist only of an input and an output layer, others build more sophisticated neural networks with a hidden layer between the other two.

1.2.1.5 Regression-based Methods

In this method the training data are represented as a pair of input/output matrices in which the input matrix is identical to our feature matrix A and the output matrix B which consists of flags that indicate the category membership of the corresponding document in matrix A. Thus B has the same number of rows as A (namely m) and c columns in which c represents the total number of categories defined. The goal of the method is to find a matrix F that transforms A into B' (by simply computing $B' = A \times F$) so that B' matches B as well as possible. The matrix F is determined by applying multivariate regression techniques.

1.2.1.6 Vector-based Methods

One of the simplest categorization methods is the centroid algorithm. During the learning stage only the average feature vector for each category is calculated and set as the centroid -vector for the category. A new document is easily categorized by finding the centroid-vector that is closest to its feature vector

SVM [135] is a very popular state-of-the art document classification technique that is used to classify documents into positive and negative classes. It is a discriminative classifier that is formally defined by a separating hyperplane. In other words, given labeled training data (labeled with predefined classes), the algorithm outputs an optimal hyperplane that categorizes new documents. The operation of the SVM algorithm is based on the finding of the hyperplane that offers the largest minimum distance to the training documents. The optimal separating hyperplane maximizes the margin of the training data. New documents are then predicted to belong to a category based on the side of the hyperplane that they fall on. Inherently, SVM classifiers are binary classifiers –which means that they classify documents into one of the two classes (yes/no.) However techniques such one-versus-one, one-versus-all can be employed to extend the SVM classifiers to multi-class document classification.

1.2.2 Knowledge Mining

Knowledge mining seeks to extract useful information from unstructured textual data through the identification and exploration of interesting patterns, phrases and concepts. It fosters strong connections with natural language processing, data mining, ML,IR and knowledge management. The field of knowledge/information mining has received a lot of attention due to the ever- increasing need for the managing of the information that resides in the vast amount of available documents. Researchers have proposed many approaches that would enable the processing of the textual content in a document in order to extract the concepts and categories from it. Below we describe some of the prominent methods of category identification by using knowledge extraction techniques.

1.2.2.1 Keyphrase-based Methods

Keyphrase extraction weighs word n-grams or syntactic phrases that appear in a document according to their statistical properties. The resulting category terms are restricted to phrases that occur in the document, and may be prone to error because semantic relations are ignored. To overcome this problem, a successful approach is to engage a controlled vocabulary such as Wikipedia (Freebase [19], MeSH[4], and DBPedia [7] are other possible vocabularies) to map the keyphrases to concepts in the vocabulary. Works done by Medelyan et al. [95] and Ferragina et al. [53] use keyphrase extraction techniques and link them to the Wikipedia titles as the topics for the document.

1.2.2.2 Clustering-based Methods

Wartena et al. [148] consider topic detection without any prior knowledge of the category structure or possible categories. Keywords are extracted and clustered on the basis of different similarity measures by using the induced k-bisecting clustering algorithm. Evaluation on Wikipedia articles shows that clusters of keywords correlate strongly with the Wikipedia categories of the articles. In addition, they find that a distance measure that is

[4] `https://www.nlm.nih.gov/mesh/`

based on the Jensen-Shannon divergence of probability distributions outperforms the cosine similarity. In particular, a newly proposed term distribution that takes co-occurrence of terms into account yields the the best results.

Petkos et al.[108] present a document-pivot algorithm for topic detection, that is, it clusters documents and treats each cluster as a topic. This work modifies a previous version of a common document-pivot algorithm by considering specific features of tweets which are strong indicators that a particular sets of tweets belong to the same cluster. It also considers the granularity of topics when performing topic detection and ranking topics.

1.2.2.3 Pattern Mining Methods

Wu et al.[152] present an approach for Topic Detection and Tracking(TDT) on the basis of the credible association rule (CAR). This paper considers topic detection without any prior knowledge of category structure or possible categories. Topic features are selected primarily on the base of CAR. This paper first uses the TF-IDF evaluation parameters to select a small amount of features as a feature pre-selection set for topic features selection and then applies the CAR and maximal cliques mining algorithm [18] in order to cluster features into topics. Topics are typically represented by a cluster of words that contribute to topic accuracy.

TopCat (Topic Categories) [36] is a technique that is used to identify topics that recur in articles in a text corpus. Natural language processing techniques are used to identify key entities in individual articles, thus representing an article as a set of items. This allows the authors to view the problem in a database/data mining context: identifying related groups of items. This paper presents a method to identify related items on the basis of traditional data mining techniques. Frequent itemsets are generated from the groups of items, followed by clusters formed with a hypergraph partitioning scheme.

1.2.2.4 NLP-based Methods

NLP produces technologies to analyze, and generate text in natural language. NLP techniques along with some statistical techniques can be employed in topic detection in

text-based documents. Carthy et al. [29, 30] present lexical chains to build effective topic tracking systems. A lexical chain is a sequence of related words in the text, spanning short or long distances. Lexical chaining is a method of grouping lexically related terms into lexical chains by using simple NLP techniques.

Nallapathi et al. [103] present a semantic language modeling approach to model news stories in the Topic Detection and Tracking (TDT) task. They build a unigram language model for each semantic class in a news story. They also cast the link detection sub-task of TDT as a two-class classification problem in which the features of each sample consist of the generative log-likelihood ratios from each semantic class. They then compute a linear discriminant classifier by using the perceptron learning algorithm on the training set.

Benhardus et al. [13] present outlines methodologies of detecting and identifying trending topics from streaming data. Term frequency-inverse document frequency analysis and relative normalized term frequency analysis are performed on the documents in order to identify the trending topics. Relative normalized term frequency analysis identifies unigrams, bigrams, and trigrams as trending topics, while term frequency-inverse document frequency analysis identifies unigrams as trending topics.

Cordobes et al. [37] present techniques that are based on graph similarity in order to classify short texts by topic. They build graphs from the input texts, and then use the properties of these graphs to classify them. The basic principle is that every piece of text (a sentence) can be represented as a graph. Essentially, for a given text this proposal uses the words in the text as graph vertices and creates weighted edges between the words. The authors have considered different ways of assessing weights on the edges. A simple option is that the weight would represents the frequency with which both words occur together in the text. Another more sophisticated (and complex) method is one in which this frequency is weighted by the distance between the words in the syntactic tree of the text. The hypothesis of the authors is that graphs that belong to the same topic have a common representative structure (topic reference graph). For text classification, they look for the similarities between the graph that is generated for a given text and different topic reference graphs.

Chen et al. [33] present algorithms for topic detection by using named entities, nouns

and verbs as cue patterns in order to relate news stories and topics. A two-threshold scheme that is proposed determines the relevance (irrelevance) between a news story and a topic cluster. Look ahead information deals with ambiguous cases in clustering. The least-recently-used removal strategy models the time factor in such a way that older and unimportant terms will have no effect on clustering. This work shows that nouns and verbs and the least-recently-used removal strategy outperform other models. The performance of the named-entity-only approach decreases slightly. However it has no overhead of the nouns-and-verbs approach with the least-recently-used removal strategy.

1.2.2.5 Interactive Methods

In order to aid the ML development, researchers have explored the visualizing of specific ML algorithms, including Naive-Bayes [11], decision trees [5], SVMs [27], and HMMs [41]. A previous study by Ware et al. has shown that such tools can produce better classifiers than automatic techniques [147]. These visualizations and interaction techniques are tied to specific algorithms. Researchers have also developed more general techniques that apply across algorithm types. Some notable techniques are highlighted below.

Godbole et al. [58] present the HIClass (Hyper Interactive text Classification) system, which is an interactive text classification system that combines the cognitive power of humans with the power of automated learners so as to make statistically sound classification decisions. HIClass is based on active learning principles and has aids for detailed analysis and fine tuning of text classifiers while exerting a low cognitive load on the user.

Talbot et al. [138] present EnsembleMatrix, an interactive visualization system that presents a graphical view of confusion matrices to help users understand the relative merits of various classifiers. EnsembleMatrix allows users to directly interact with the visualizations in order to explore and build combination models. They evaluate the efficacy of the system and the approach in a user study. Results show that users are able to quickly combine multiple classifier that operate on multiple feature sets in order to produce an ensemble classifier with an accuracy that approaches the best-reported performance classifying images in the CalTech-101 dataset.

1.2.3 Topic Modeling

Topic models provide a convenient way to analyze large quantities of unclassified text. A topic contains a cluster of words that frequently occur together. A topic model can connect words with similar meanings and distinguish between uses of words with multiple meanings. In this section, we highlight some of the well known techniques for topic identification in the topic modeling paradigm.

1.2.3.1 Latent Semantic Analysis (LSA)

Dumais et al. [50] present LSA[5], a technique in natural language processing, (distributional semantics, in particular,) of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts that are related to the documents and the terms. LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix that contains word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique that is known as singular value decomposition (SVD[6]) is used to reduce the number of rows while the similarity structure among columns is preserved. Words can then be compared by taking the cosine of the angle between the two vectors (or the dot product between the normalization of the two vectors)that are formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

1.2.3.2 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (PLSA)[7] [64], which is also known as probabilistic latent semantic indexing (PLSI) is a statistical technique that analysis the two-mode and co-occurrence data. In effect, one can derive a low-dimensional representation of the observed variables in terms of their affinity to certain hidden variables, (as is seen in LSA,)from which PLSA evolved.

[5] https://en.wikipedia.org/wiki/Latent_semantic_analysis

[6] https://en.wikipedia.org/wiki/Singular_value_decomposition

[7] https://en.wikipedia.org/wiki/Probabilistic_latent_semantic_analysis

Compared to standard LSA which stems from linear algebra and downsizes the occurrence tables (usually via a singular value decomposition), PLSA is based on a mixture decomposition that is derived from a latent class model. The main goal of PLSA to identify and distinguish between different contexts of word usage without recourse to a dictionary or thesaurus. It includes two important implications: Firstly, it allows for disambiguation e polysemy,(that is, words with multiple meanings). Secondly, it discloses topical similarities by grouping words that share a common context.

1.2.3.3 Latent Dirichlet Allocation (LDA) and Its Derivatives

LDA [17] is a technique in text mining which is based on statistical (Bayesian) topic models and is very widely used. LDA is a generative model that in some sense tries to mimic the writing process. In LDA, each document is modeled as a mixture of topics, and each topic is a discrete probability distribution that defines how likely each word is to appear in a given topic. These topic probabilities give a concise representation of a document. This is similar to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a Dirichlet prior. In practice, this results in more reasonable mixtures of topics in a document. In these techniques, a "document" is a "bag of words" with no structure beyond the topic and word statistics.

Researchers have proposed several other models that are based on LDA for specific scenarios; Some of the popular models include finding bursty topics [45], on-line LDA [4], author-topic analysis [132], supervised topic models [94, 162, 113], Latent Dirichlet coclustering [126] and time variant topic models [146].

1.2.3.4 Correlated Topic Model

The LDA model assumes that the words of each document arise from a mixture of topics, each of which is a distribution over the vocabulary. A limitation of LDA is the inability to model topic correlation even though, for example, a document about genetics is more likely to be also about disease than X-ray astronomy. This limitation stems from the use of the Dirichlet distribution to model the variability among the topic proportions. In

the correlated topic model [16], the topic proportions exhibit correlation via the logistic normal distribution [3]. The authors derive a mean-field variational inference algorithm for approximate posterior inference in this model. However, this becomes complicated by the fact that the logistic normal is not conjugate to the multinomial. The authors show that gives a better fit than LDA on a collection of OCRed articles from the journal Science and provides a natural way of visualizing and exploring this unstructured data set and others.

1.2.3.5 Hierarchical Topic Models

Hierarchical Topic Models learn to organize the topics according to a hierarchy in which more abstract topics are near the root of the hierarchy and more concrete topics are near the leaves. Several hierarchical topic model techniques have been proposed by the researchers, which are highlighted below.

One of the popular hierarchical topic models, hLDA [15], is based on the Nested Chinese Restaurant Process (NCRP [15]). In NCRP it is assumed that there are an infinite number of infinite-table Chinese restaurants in a city. One restaurant is identified as the root restaurant, and on each of its infinite tables is a card with the name of another restaurant. This structure repeats infinitely many times. Each restaurant is referred to exactly once. The restaurants in the city are organized into an infinitely branched, infinitely-deep tree. Note that each restaurant is associated with a level in this tree. The root restaurant is at level 1, the restaurants referred to is on its tables; cards are at level 2, and so on. hLDA generate a mixture distribution on topics using an NCRP prior. Topics are joined together in a hierarchy by using the NCRP. The generative process picks a topic according to their distribution and generates words according to the word distribution for the topic.

Hierarchically Supervised LDA (HSLDA) [107] takes advantage of hierarchical supervision (that is labels). The HSLDA model is based on the intuition that the hierarchical context of labels provides valuable information about labeling. HSLDA jointly models documents and responses by drawing response variable realizations from a normal distribution, and generates label responses using a hierarchy of conditionally dependent probit regressors

[107]. In the joint modeling of each document, both empirical topic distribution and whether the parent label is applied to the document determine whether a label is to be applied. The HSLDA model views word-multinomials (topics) as global constructs and links them to hierarchy nodes through per-label topic distributions.

Slutsy et al. [128] present a Tree Labeled LDA (tLLDA) algorithm to infer topic by models using its manually compiled ontology, DMOZ. The algorithm takes advantage of the hierarchical nature of the DMOZ ontology and infers topic models by jointly modeling word and ontology node assignments for documents. More precisely, the tLLDA model estimates a single word-multinomial for each node of the target ontology.

The Pachinko Allocation Model (PAM) [85] captures arbitrary, nested, and possibly sparse correlations between topics using a directed a cyclic graph (DAG). The leaves of the DAG represent individual words in the vocabulary, while each interior node represents a correlation among its children, which may be words or other interior nodes (topics). PAM provides a flexible alternative to Correlated Topic Models (CTM) [16], which captures correlations only between pairs of topics.

1.2.3.6 Non-Negative Matrix Factorization (NMF)

A highly-effective alternative to LDA is to use Non-NMF [82]. NMF refers to an unsupervised family of algorithms from linear algebra which simultaneously perform dimension reduction and clustering. Given a non-negative input matrix, which describes a set of items using a fixed number of features, NMF produces a factorization of the input matrix to reveal the meaningful " latent features" that are hidden in the data. Each item can be viewed as being built up from these latent features, and NMF allows us to identify clusters of items that share the same latent features. In the context of textual data, in which our items are documents, the input matrix will be a document-term matrix that describe the frequency of each term (word) in each document. The goal of NMF is then to uncover the latent features that correspond to the themes or topics that are most prominent across the entire corpus [61].

1.3 Our approach

Next we outline our approach of building a categorization system addressing the challenges stated earlier. We start with the motivation for building such a system (in fact we highlight what are the shortcomings in the existing systems) and thereafter outline our techniques.

1.3.1 Motivation

The motivation for our work, which is the adaptive organizing of digital documents, comes from the need for effective categorization systems in order to organize, search and extract information, and address the challenges that are described in Section 1.1. Although vast amount of literature on document classification, topic modeling and knowledge extraction techniques exist, no single technique addresses all the challenges listed in Section 1.1.

The supervised document classification techniques are highly effective and widely used. However, they require a good amount of training data and a predefined set of class labels. As the number of class labels increases, the effort to prepare the training data increases and soon becomes prohibitively expensive. Moreover, coming up with the right number of class labels is a bigger challenge. None of the document classification algorithms described in Section 1.2.1 address this problem. They assume that the labels and training data are predefined.

On the other hand, topic modeling techniques attempt to identify the topics from the dataset, in an unsupervised manner. Since the topics are represented as a collection of words, additional effort is required in order to produce semantically and grammatically meaningful names for these topics. Although the topic names can be produced in an automated fashion [81] often-times they need manual correction. The topics discovered by the topic models are sensitive to the data; when more documents are added to the dataset, the topics (that is, word distributions over the topics) change. Moreover, topic models often produce word distributions that are difficult to interpret. For example, in our experiments in creating categories for a document collection, LDA produced many topics such as {report , official, said , ask , told , made , comment , time, way, come, try,... }, which could not be assigned any meaningful title/label.

Knowledge mining approaches (Section 1.2.2) have shown good promise in identifying per-document categories. Combining the local (per-document) category information with the global (corpus) categories is a challenging task. Simply collating the categories from each document leads to over-specified organization.

In order to address the challenges of document organization (listed in Section 1.1) and to leverage the strengths of various categorization/classification/topic identification methods, we propose a framework that can evolve an adaptive document organization technique. Inspired by the works of Medelyan et al. [95] and Ferragina et al. [53], coupled with advances in the collaboratively built Knowledge Bases (such as Wikipedia, DMOZ, DBPedia, and the like) we propose a multistage document organization methodology with each stage solving a specific sub-problem and making use of the knowledge bases in these stages to achieve desired goal. The framework that we introduce can incorporate different types of document classification, topic modeling and knowledge mining techniques as features. It can also learn a suitable linear mixture of them. The framework is also capable of incorporating user preferences and adapting to the changes in the category distribution in a growing collection of documents.

The availability of a right Knowledge Base is very important for our approach to work. The Knowledge Base must be exhaustive enough to cover the vocabulary of the document collection. Today, Knowledge Bases such as Wikipedia, Ohsumed, DBPedia, Linked Open Data, and the like have become popular. They contain enormous amount of information that is curated by human contributors.The effective utilization of these Knowledge Bases can offer many benefits in the organizing of digital documents. (i) The concepts in these Knowledge Bases can help to organize the digital artifacts. (ii) Rich context is available for each concept in the Knowledge Base which makes the organization of digital documents more meaningful. (iii) The vast coverage of concepts in these Knowledge Bases can encompass the terminology of nearly any digital collection. (iv) These knowledge bases keep growing in new concepts from the contributions of the community members.

1.3.2 Multistage Document Organization

Our multistage document organization process is depicted in Figure 1.1. Each rectangular box represents a stage that solves a specific sub-problem of the bigger problem - document organization. The functionality of each of these stages is briefly described below and discussed in detail in subsequent chapters.

1.3.2.1 Associating Concepts with Digital Documents Using a Knowledge Graph

At this stage, using a novel technique, concepts from a knowledge graph are associated with the digital documents. We construct a knowledge graph from Knowledge Bases such as Wikipedia by creating the concept nodes and relationship edges between them. For each digital artifact (document, video transcript, and the like) in the collection, we construct an AMN of concepts from the knowledge graph. The AMN node potentials measure how good a concept node is for the digital artifact, and the edge potentials measure the strength of the relationship between the concepts. The static and dynamic node potentials of AMN can incorporate various measures such as document similarity (Cosine, Jaccard, BM25, and the like), decision values from various classifiers (SVM, decision trees, and the like), topic model affinity and user preferences. The edges help to share the information between the nodes. A MAP inference that uses binary node labels gives the association of the concept nodes to the digital artifacts. We also present techniques to prune the irrelevant nodes from the AMN by using query and language models to improve the inference time.

1.3.2.2 Active Learning for Document Organization

In order to learn user preferences, it is important to provide feedback to the system. This feedback tells the system whether it has associated right concepts from the knowledge graph with the digital artifacts. Based on this feedback, the system learns a model so as to incorporate user preferences. In order to receive feedback on the most uncertain

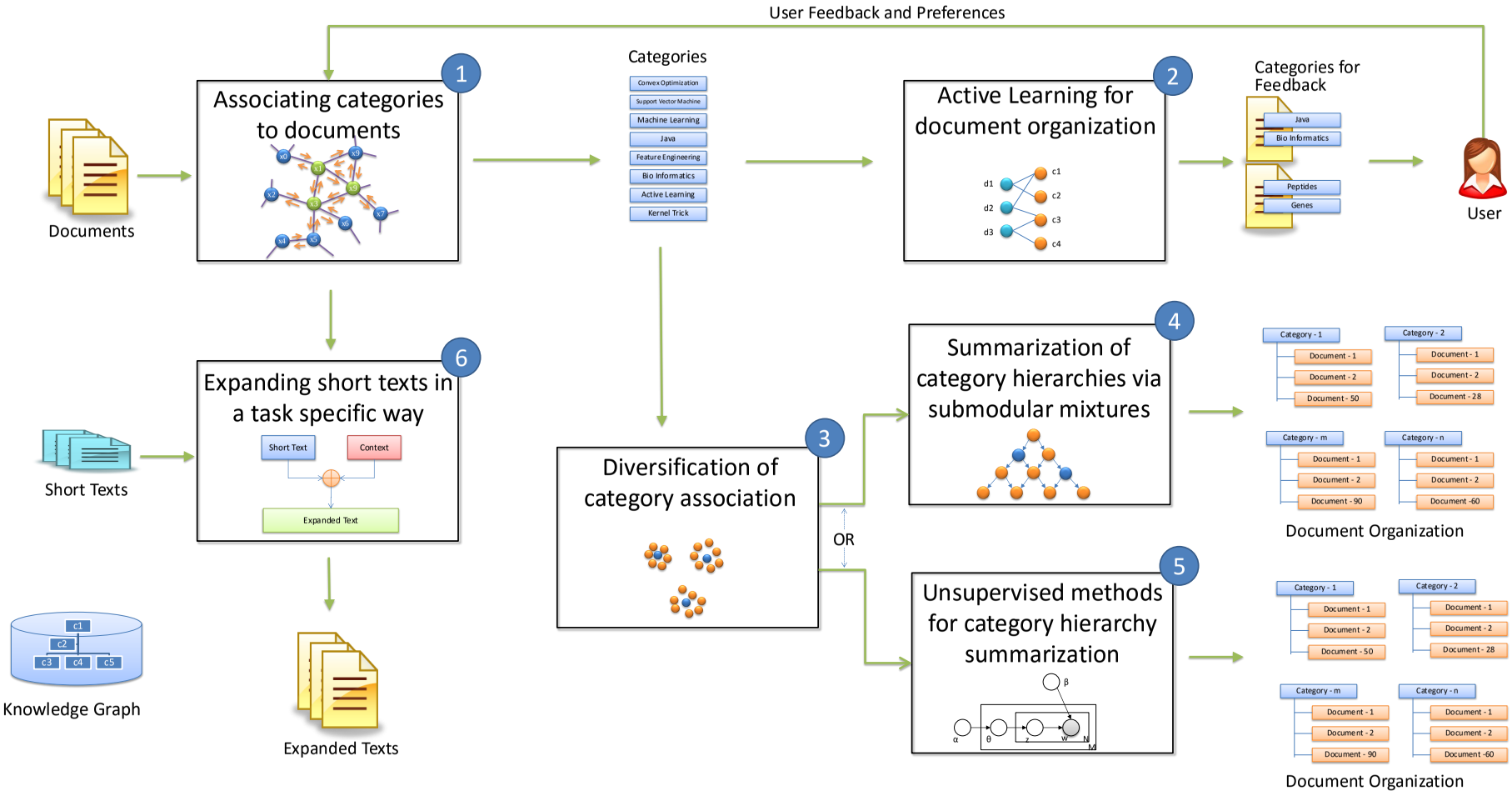


Figure 1.1: Adaptive document organization process

cases of concept association,we develop uncertainty sampling techniques for active learning. We show that our AMN inference problem can be converted to a margin based classification problem by which the notion of distance from the margin can be realized for the uncertainty sampling principle of active learning. In addition, we also present joint active learning in the document and the concept space in order to seek feedback on most uncertain pairs of document-concept association.

1.3.2.3 Diversification of Concept Association

Due to the associative property of AMN, a digital artifact usually gets associated with many similar concepts. A post-facto diversification of concepts that are associated with an artifact is needed to (i) get rid of very similar concepts, and (ii) retain a diverse set of concepts in an artifact. It is natural to diversify the concepts using the concept graph that is used for AMN because it already possesses the required features and signals. We developed a diversification technique that is based on bi-convex optimization of the relevance and the divergence of concepts that are associated with a digital artifact. Our experimental results show that in diversification, the technique performs better than several baseline techniques.

1.3.2.4 Summarization of Concept Hierarchies via Submodular Mixtures

In order to address the problem of proliferation of fine grained concepts that are associated with the digital collection, we study the problem of summarizing concept hierarchies over a given set of artifacts. Unlike previous works, we directly pose the problem as a submodular optimization problem on a concept hierarchy by using objects as features.The desirable properties of the chosen concepts include object coverage, specificity, concept diversity, and concept homogeneity, each of which, we show, is naturally modeled by a submodular function. Other information, that is provided for example,by unsupervised approaches such as LDA and its variants, can also be utilized by defining a submodular function that expresses coherence between the chosen concepts and this information. We use a large-margin framework to learn convex mixtures over the set of submodular components. We empirically evaluate our method on the problem of organizing articles for Wikipedia

disambiguation pages using human generated clustering as ground truth. We find that our framework improves upon several baselines according to a variety of standard evaluation metrics such as the Jaccard Index, F1 score and NMI. Moreover, our framework can be scaled up easily to extremely large scale problems.

1.3.2.5 Unsupervised Methods for Concept Hierarchy Summarization

The previous section presented a technique to generate a set of summary concepts in a supervised learning framework. It needed a labeled training data in order to learn the submodular mixture. We used the Wikipedia disambiguation pages as our training data. However, in other collections, it may become laborious to prepare the training data. In order to overcome this difficulty, we explore other techniques in an unsupervised setting so that a DAG structured summary of concepts that can best describe the collection can be learned. Given a collection of artifacts with fine-grained concept assignment, and a massive concept hierarchy in the form of a DAG, we infer a sub-DAG as a summary DAG that has the maximum likelihood of generating the collection. The concept DAG is modeled as a Markov Network in order to generate the artifacts in the collection. It specifies a joint probability distribution over the observed artifacts in the collection. We learn this distribution through Gibbs sampling. Thereafter we estimate the importance of concepts that are based on the marginal probabilities of the concept nodes and its children, in order to rank the concepts. The top K ranked nodes along with the edges between them form a sub-DAG. As in the previous case, we evaluate this method on the Wikipedia disambiguation pages.

1.3.2.6 Expanding Short Texts in a Task Specific Way and Organization

We present a method for the expanding of short texts using a "universal corpus" (such as Wikipedia, Ohsumed) of documents, for the better organization of short texts. Our method uses a derivative free algorithm called BOBYQA to iteratively reduce the loss function of the task using the short texts. In each iteration, an expansion candidate is generated and the task is evaluated with the candidate expansion text. Based on the result of the evaluation, a quadratic approximation of the loss function is computed and

further minimized using the trust region technique. This makes each iteration improve the short text expansion. As a result, we learn a model to expand the short text by mapping it to the best possible article from the " universal corpus" . Our experimental results on the short text classification task show that our method improves over several baselines and existing methods.

1.4 Demonstration of Document Organization using Our Approach

In this section we demonstrate the application of our method to an automatic document organization problem. For this, we extracted approximately 150 technical articles under science tracks from 10 subjects (Computer Science, Chemistry, Computational Biology, Micro Biology, Genetics, Physics, Electricity, Logic (Mathematics), Algebra, and Number Theory) from different sources (DOAJ and arXiv). We associated categories for each article using our AMN framework (Block 1 in Figure 1.1; Chapter 2). Wikipedia was used as Knowledge Base during this experiment. While associating concepts from the Wikipedia with the articles, we also added user preferences to limit the categories to only those under science in Wikipedia (constraints were added for Arts, Music like categories and their descendant categories in the Wikipedia category hierarchy). With the help of our proposed active learning technique (Block 2; Chapter 3) we provided some feedback to the system so as to improve the accuracy of categorization. We then applied our diversification algorithm to the associated categories so that each document was associated with 20 categories (Block 3; Chapter 4) at the most. Thereafter, using our proposed submodular category summarization technique (Block 4; Chapter 5), we summarized to 100 categories (we used weights learned from our earlier disambiguation page generation experiments that are described in Chapter 5.) The Figure 1.2 shows the summarized 100 categories in the form of a word cloud. The font size of the category names indicates the number of documents organized under them. (Color does not indicate anything.)

The major categories under which documents were organized were Genetics, Physics, Chemistry, Mathematical-Science, and the like. Our careful observation of the documents in the collection and the categories that were discovered by our approach shows that,

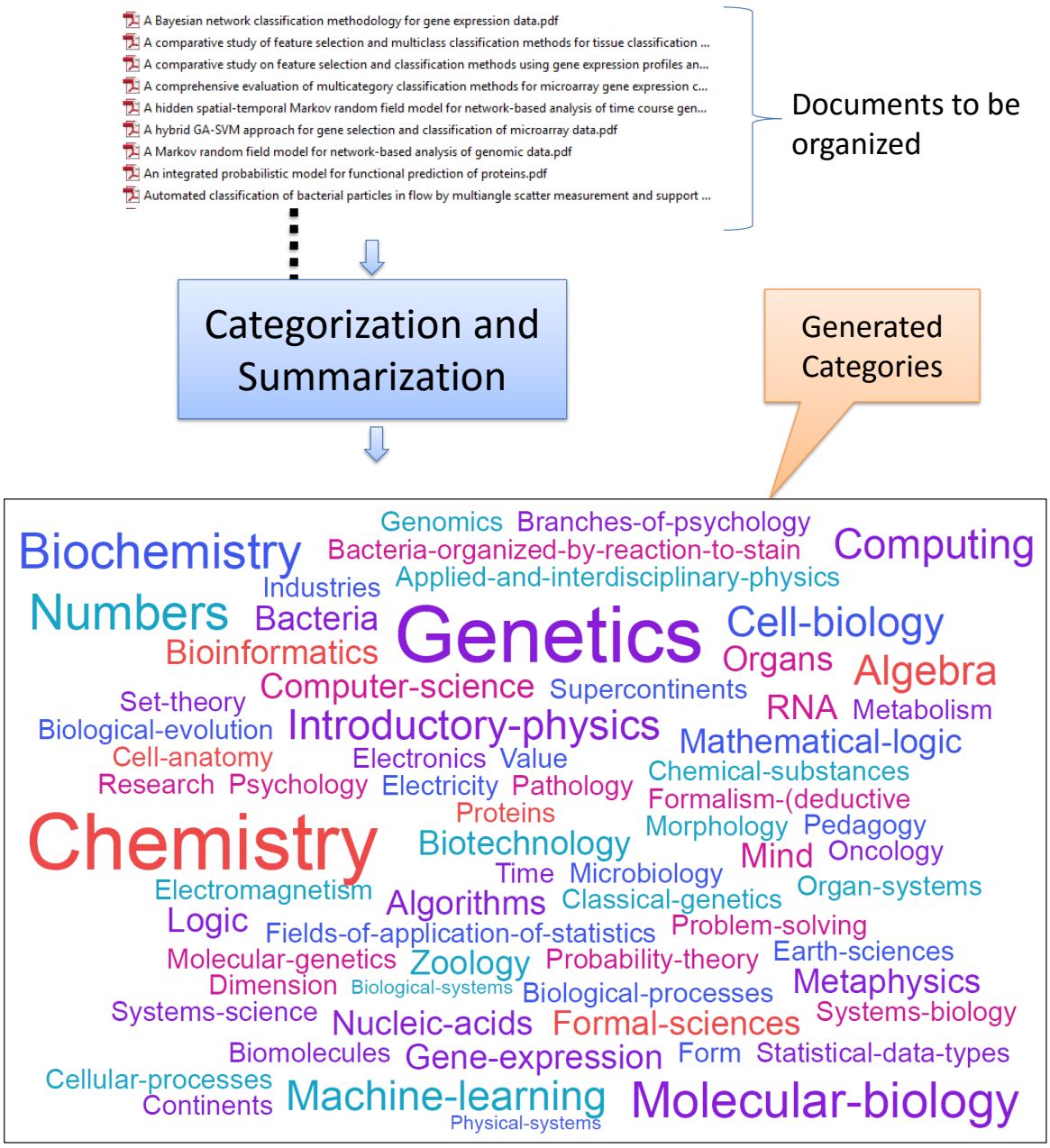


Figure 1.2: Categories discovered by our method from the collection of technical articles

these categories represent the documents in the collection very well.

In order to compare quantitatively, we adopt the consistency measure used by Rolling et al. [116] as follows:

$$Consistency = \frac{2|A \cap B|}{|A| + |B|}$$

where A and B are the categories/topics that the two systems assign.

We took top 20 categories suggested by our system for consistency computation. Since we formed the document collection by sampling the documents from 10 different categories, we know the *true* categories for this document collection. We then manually went through the selected 20 categories from our system and marked them whether they correspond to any of the true categories or not. We could now compute the consistency between the categories discovered by our method and the true categories using the above formula. We observed a consistency of 75%.

1.4.1 Comparison with LDA topics

LDA being one of the predominantly used techniques in discovering topics from a collection of text documents, it is an interesting experiment to compare the outputs of our technique with LDA. Since LDA topics are probability distribution over words, it becomes challenging to compare the outputs. We adopt three simple strategies to overcome this problem:

1. We run LDA on 150 documents and generate 20 topics. We then extract the keywords from the LDA discovered topics and project them as the categories for those documents. The keywords are the high probability words from the topics. Figure 1.3 shows the word cloud of top 100 categories (keywords), drawn using aggregate probabilities of the keywords across the topics as the font size. Visual inspection and comparison of this word cloud with the one generated from our method (Figure 1.2) shows that the categories suggested by our method appear reasonably closer

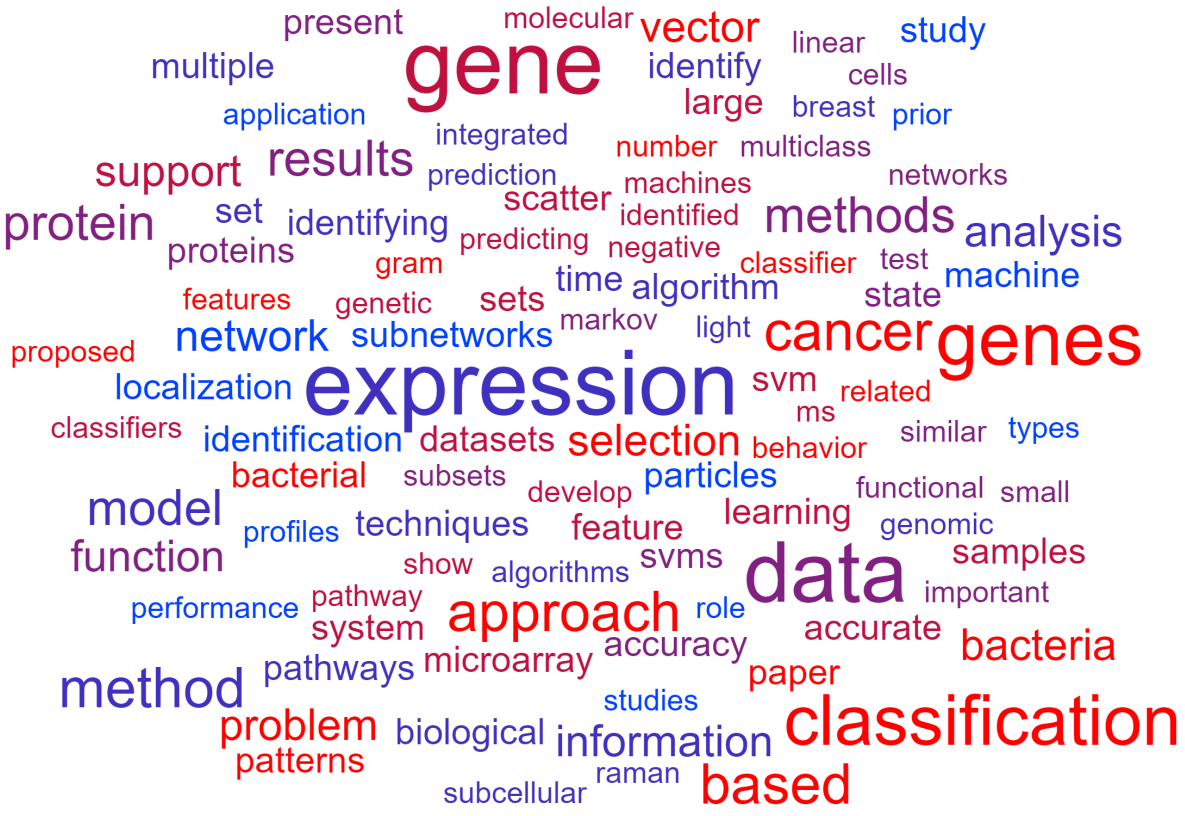


Figure 1.3: Keywords from the topics detected by LDA from the collection of technical articles

to the *t*rue categories (categories from which we had sampled the documents) than the LDA suggested categories.

As in the previous case, to compute the consistency metric, we took the top 20 categories (as described above) from the LDA topics and marked them whether they correspond to any of the true categories or not. We observed that the consistency between the LDA categories and the true categories is 50%.

We believe that the consistency is low because of the following reasons:

- While the LDA groups coherent topical terms together, each term is able to express some characteristics of the topic, and hence cannot stand by itself as a topic always. Meanwhile in our method, the Wikipedia guided fine-grained category assignment and category summarization is able to capture better semantic categories and their labels.
- Aggregating keyword probabilities from the LDA topics is essentially equivalent to finding high frequency words from the document collection. Projecting these

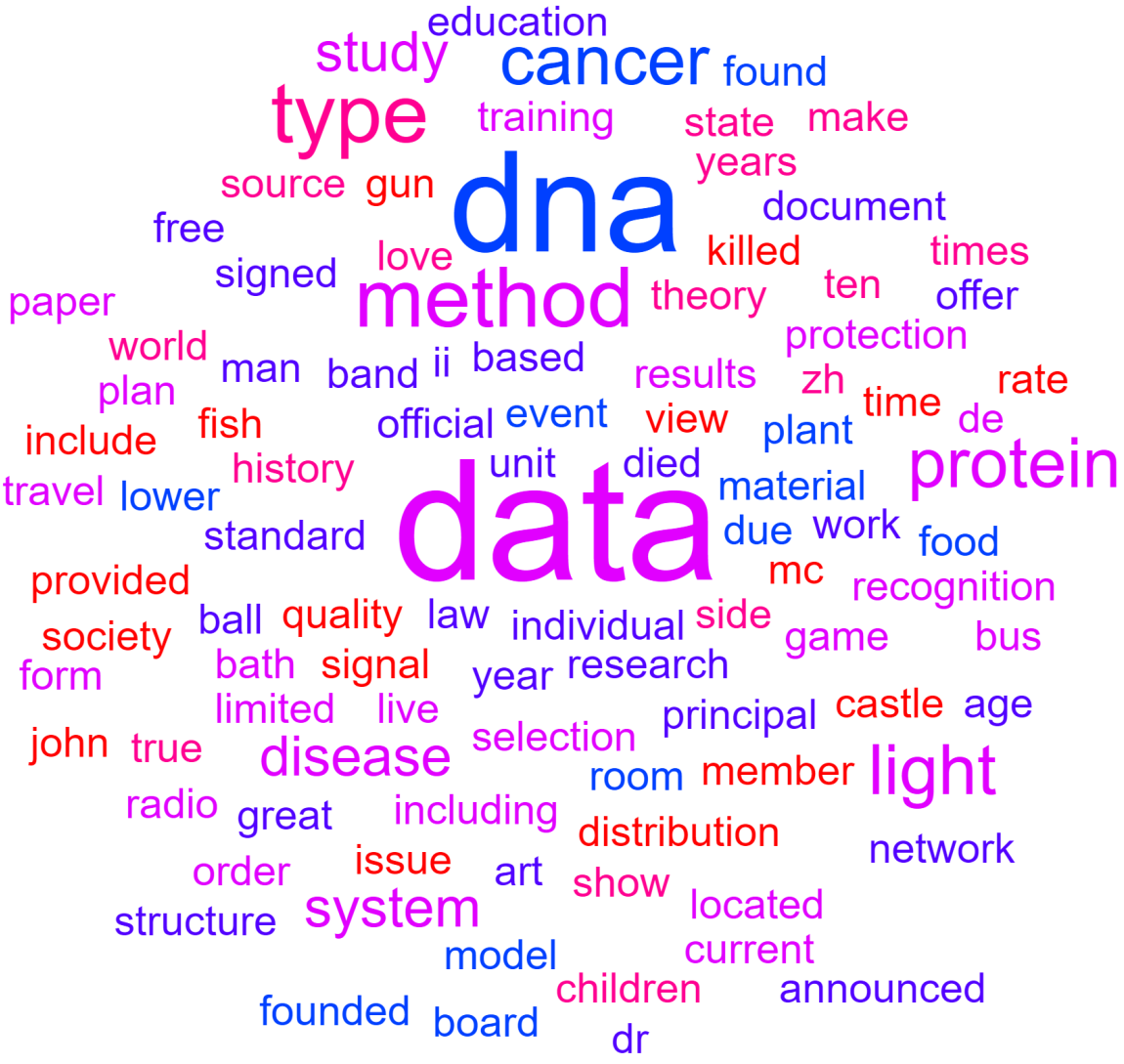


Figure 1.4: Keywords from the topics detected by LDA from the collection of technical articles along with 1M Wikipedia articles

words as the categories may not necessarily be a good choice of categories for the entire collection.

- The collection having merely 150 documents is not large enough for the LDA algorithm to discover 20 topics. (In the next experiment we address this issue by running LDA on 1 million Wikipedia articles.) On the contrary, our method is able to discover the categories for a small collection, and as well as a large collection. This happens due to the fine-grained categories being summarized to the right level of generalization by considering the coverage of documents in the collection.

2. One of the drawbacks of the previous experiment is that the size of the document collection used for running the LDA is very small (i.e, 150 documents). To overcome this problem, in this experiment, we included about one million Wikipedia articles sampled randomly from 5M articles and then ran LDA on it. The number of topics for LDA was set to 1000. We then took the topics assigned to 150 of our DOAJ/arXiv documents and discovered the categories for them exactly as done in the last experiment. Figure 1.4 shows the word cloud drawn for the categories discovered by this method. We observed a consistency of about 45% in this case.

   Below we discuss our observations on these results:

   - Similar to the observations done in the previous case, top probability word from the LDA topic alone is not sufficient to make up a good category name.
   - Since we included 1M Wikipedia articles while running the LDA, the word distribution for the topics predominantly comes from the Wikipedia articles. Due to this, the top word of a topic may not be entirely relevant to the small set of 150 DOAJ/arXiv document collection. We do not encounter such problems in our proposed method, since it operates at granular level while assigning categories to the documents and then summarizes it to the right level.

3. In this method, we use 10 high probability words from each of the LDA topic (discovered by running LDA on 150 DOAJ/arXiv documents, with 20 topics) to form a term-weighted Lucene query and search a Lucene index that we created using the Wikipedia 2012 articles dump. The title of the first article returned by the search is used as the topic name. We call this method as the *Lucene-first* method.

   Similar to the previous cases, we draw in Figure 1.5 a word cloud of these categories by taking the Lucene assigned score to the top result. Again, visual inspection and comparison to the word cloud from our method indicates that the categories suggested by our method matches much more closely with the true categories than the Lucene-first method. The consistency of the Lucene-first discovered topics with the true categories is about 35% – indicates that this method is not effective. We observed that in the most of the cases, the Lucene query returns fine-grained article names as the categories, since they match most of the query terms. Since more

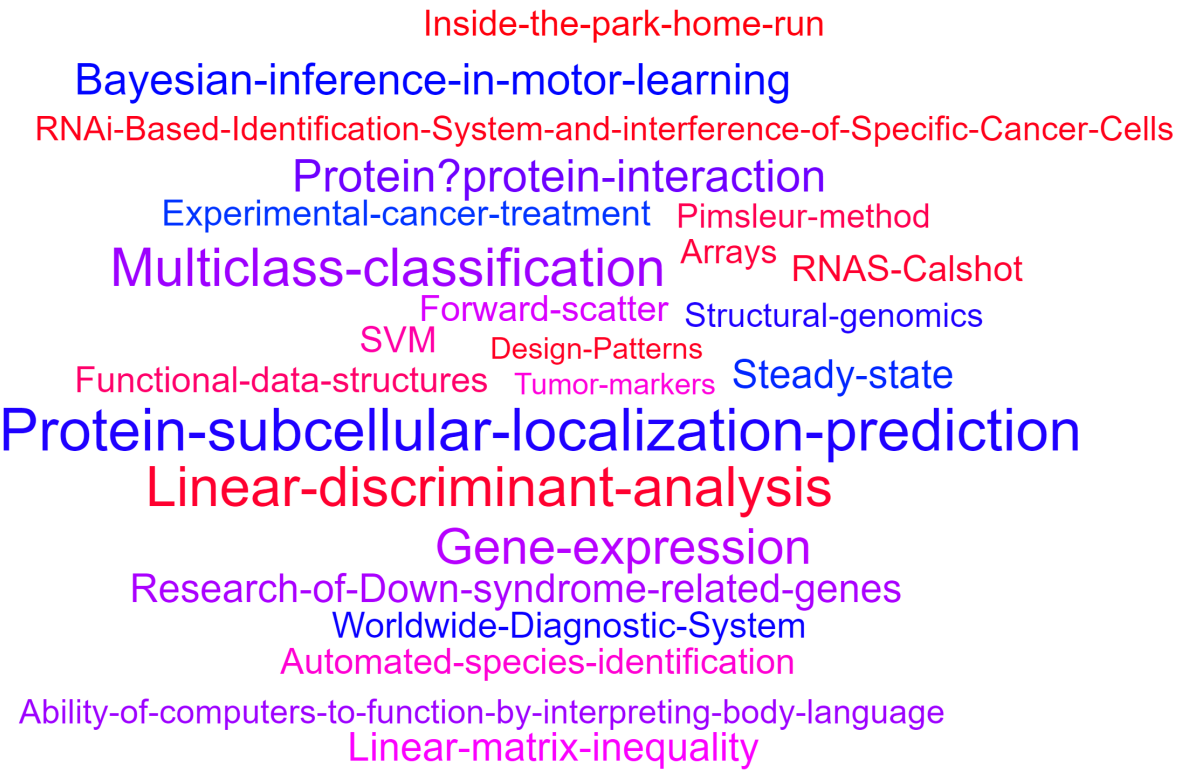


Figure 1.5: Lucene-first article names (as categories) obtained by querying Lucene index of Wikipedia articles with term-weighted query by taking high probability 10 words from topics detected by LDA from the collection of technical articles

generic articles (such as Physics, Chemistry) do not contain all those detailed terms from the LDA topic, they fail to figure in the top search result. Meanwhile, in our method, the summarization step effectively generalizes the fine-grained categories into generic ones.

1.4.2 Comparison with Lau et al. work

One of the well known work on automatic labeling of topic models is by Lau et al. [81]. In this work authors develop a system for associating LDA topics with Wikipedia article titles that act as textual (and human interpretable) labels for the LDA topics. We attempted to implement the system described by the authors in our setting and compared the results with our technique along similar lines to the previous cases (described in section 1.4.1), using the consistency measure. Due to lack of training data, and non-availability of some of the sub-systems, we were not able to get good quality results from the Lau

et al. approach. We observed a consistency value of 35% for the discovered topics. In addition, like in the previous cases, the system generated fine-grained article names as the topic names, which failed to capture the generic topics (such as Physics, Chemistry, etc.). Meanwhile, our method, due to the summarization step, effectively generalized the fine-grained categories into generic categories.

1.5 Other Applications of Our Research

Apart from document organization, the algorithms and framework that were developed as part of our research can be applied to numerous other applications.Some of these applications are highlighted below.

1.5.1 Automatic Table of Content Suggestion

Although we confine our thesis mainline to the organization of digital documents, we think that the proposed techniques can also be helpful in other types of settings such as automatic suggestion of table of contents (ToC) for a collection of chapters. In this case, we can reformulate the problem as the organizing of chapters under summarized categories as a ToC. Each chapter has a rich context (section headings, description text, and the like.), which can be used to associate sections/chapters to various concepts from the Knowledge Base. Summarizing these concepts at different levels can generate a ToC.

1.5.2 Wikipedia Style Disambiguation Page Generation

Given a collection of articles that span different categories, but with similar titles, automatically generate a disambiguation page for those titles by using Wikipedia category hierarchy[8] as a category DAG. Disambiguation pages[9] on Wikipedia are used to resolve conflicts in article titles which occur when a title is naturally associated with multiple articles on distinct categories. Each disambiguation page organizes articles into several

[8] `http://en.wikipedia.org/wiki/Help:Categories`

[9] `http://en.wikipedia.org/wiki/Wikipedia:Disambiguation`

groups, in which the articles in each group pertain only to a specific category. Disambiguation may be seen as paths in a hierarchy leading to different articles that arguably could have the same title. For example, the title Apple[10] can refer to a plant, a company, a film, a television show, a place, a technology, an album, a record label, and a news paper daily. The problem then, is to organize the articles into multiple groups in which each group contains articles that are similar (categories) and has an appropriately discerned group heading. This can be achieved by applying our techniques (concept association via AMN framework (Chapter 2) and topic summarization using submodular framework (Chapter 5)

1.5.3 Refinement Search and Navigation

It is very common to show "tags" along with search results (for example Google news.) These tags help in providing the broad category overview of underlying search results or articles. They guide the user to refine their search by following those tags. By applying our technique, we can generate tags that honor various quality criteria that are discussed in our thesis.

1.5.4 e-Catalog Creation

Electronic catalogs (e-catalogs) in a digital library yield the categorical segregation of articles and help the user to drill down a category hierarchy to search an article. Our technique can identify e-catalog categories. We can recursively apply our technique to generate category hierarchy on the fly. To start with, our technique can present top K topics as the first level of categories in the e-catalog. After the user chooses a particular category, we can dynamically regenerate further K topics by considering only the documents that are categorized under the chosen category.

1.5.5 Tag Cloud Generation

A Tag Cloud pictorially shows the K topics of the underlying corpus by using different font sizes. Our technique can generate those tags meaningfully (honoring various quality

[10] `http://en.wikipedia.org/wiki/Apple_(disambiguation)`

criteria.)

1.6 Contributions of This Thesis

The major contributions of this thesis are the development of:

- A framework for the evolution of an adaptive document organization approach that can:
  - address the document organization challenges that are described in Section 1.1, that is, under/over-specified categorization, intent coverage and temporal relevance of document organization:
  - incorporate outputs of different types of document classification, topic modeling and knowledge mining techniques as features and learn a suitable linear mixture of them:
- A generic algorithm –that is based on AMN – order to enable matching, such as documents and topics, news and advertisements, videos and course contents, and so on.By applying this algorithm we demonstrate how every digital artifact is associated with the relevant concept nodes from the knowledge graph. In addition to identifying the right concept nodes in order to organize the collection, we also consider user interests in the identification of the concept nodes. Users may input their preferences and provide feedback to the model/system.
- Active learning techniques that enable the finding of the most uncertain document and category pairs in an AMN. With the help of these techniques, we demonstrate the identification of the most uncertain document-category pairs in the seeking of user feedback. This feedback is then added as constraint in the AMN framework so that future category identification honors user preferences and restrictions.
- A diversification technique using a knowledge graph. The application of this technique in order to diversify the associated categories to the documents is presented in this thesis. Apart from this application, the technique is also applicable to other

settings such as the diversification of search results, diversification of sentences in a summarization task, and the like.

- A learning framework for the summarizing of a DAG-structured category hierarchy by using submodular mixtures. Unlike that which is seen in previous works, we directly pose the problem as a submodular optimization problem on a concept hierarchy by using documents as features.The desirable properties of the chosen concepts include document coverage, specificity, concept diversity, and concept homogeneity, each of which, we show, is naturally modeled by a submodular function.

- A method (based on EM and Gibbs sampling) that enables the generation of the most representative DAG of concepts (or categories) that represent the document collection. Given a collection of artifacts with fine-grained concept assignment, and a massive concept hierarchy in the form of a DAG, we show how to infer a sub-DAG as a summary DAG that has the maximum likelihood of generating the collection.

- A short text expansion technique that can be applied on several ML, NLP and IR tasks on short texts (such as short text classification, clustering, entity disambiguation, and the like) without using task specific heuristics and domain-specific knowledge for expansion. At the same time, our technique is capable of learning to expand short texts in a task-specific way. That is, the same technique that is applied to expand a short text in two different tasks is able to learn to produce different expansions depending upon what expansion benefits the task's performance.

1.7 Organization of This Thesis

The rest of the thesis is organized as shown in Figure 1.6

Chapter 2: Describes a framework to associate concepts from a knowledge graph with documents.

Chapter 3: Introduces methods for identification of most uncertain document-category pairs in order to seek user feedback in a uncertainty based active learning setup.

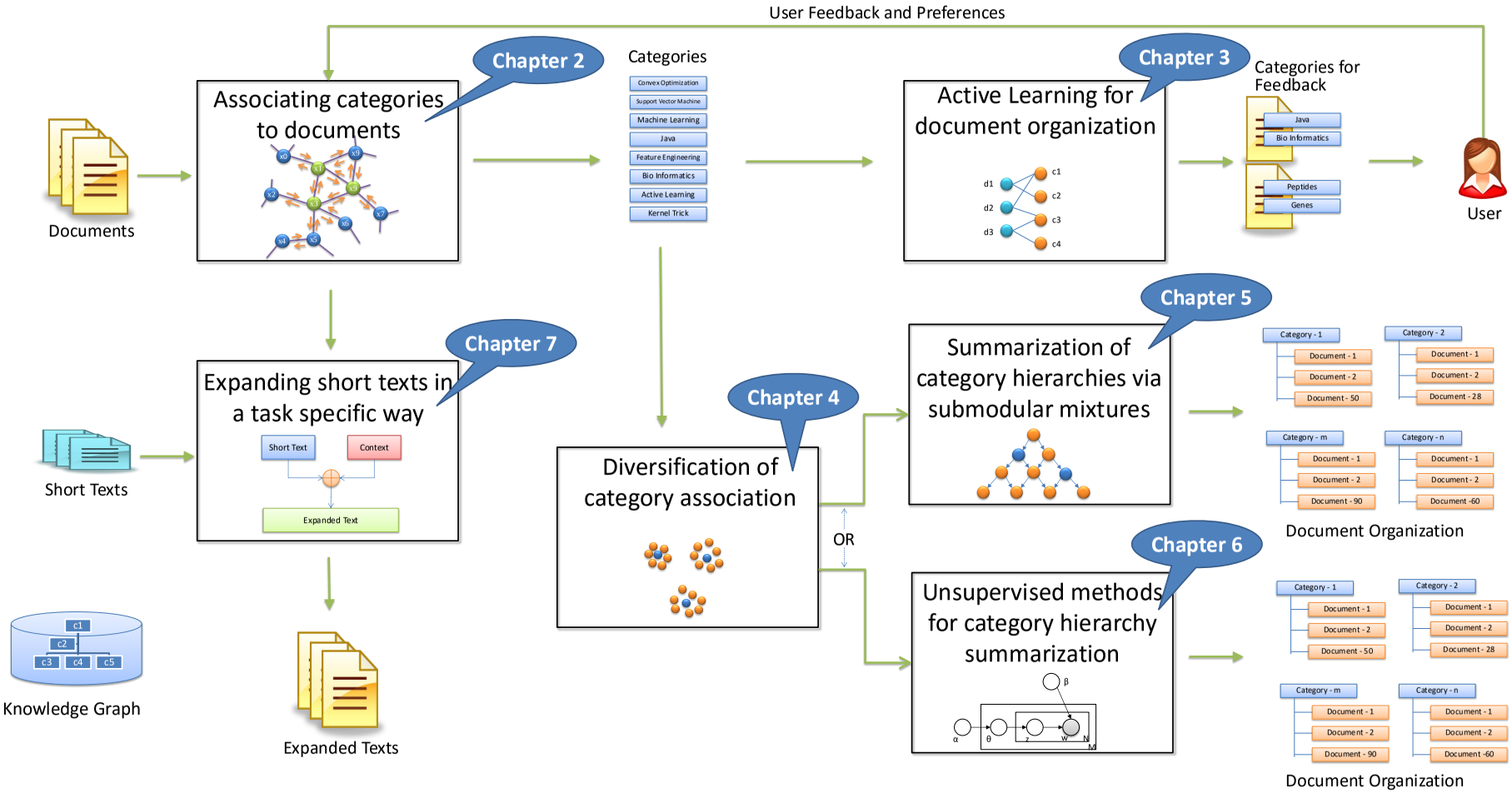


Figure 1.6: Organization of this thesis

Chapter 4: Presents a method for diversification of category assignment to the documents using a knowledge graph.

Chapter 5: Describes a framework to summarize a DAG-structured category hierarchy by using submodular mixtures. This framework allows us to create corpus (document collection) level categories from the per-document categories, by using the summarization technique.

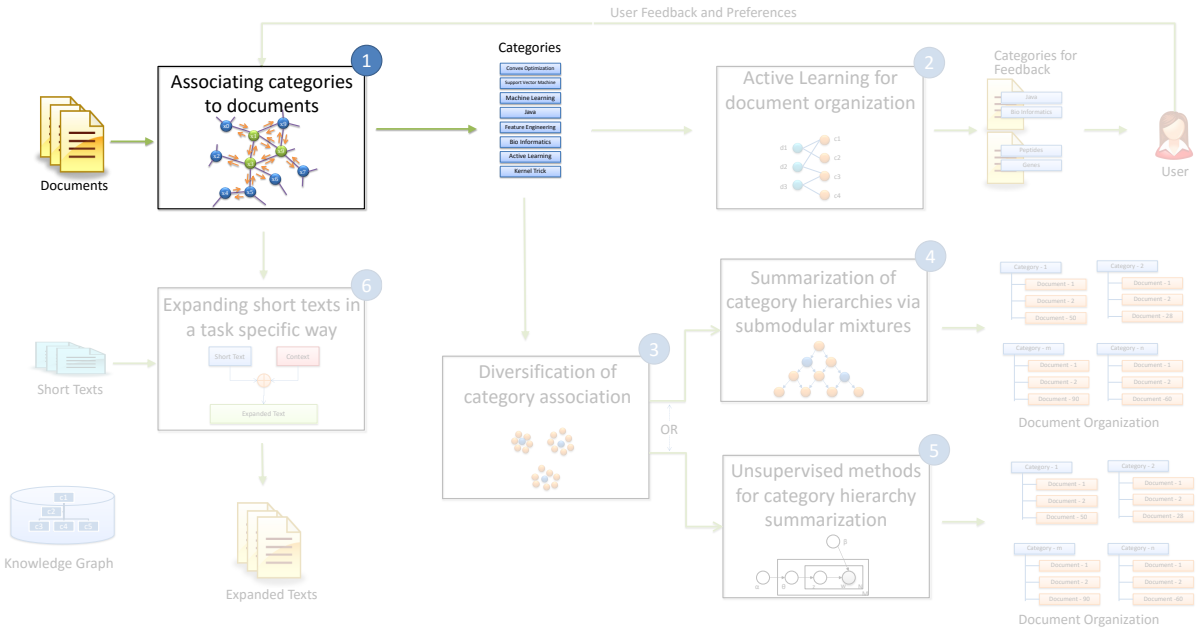
Chapter 6: Presents a DAG-structured organization of documents through unsupervised methods- EM and Sampling techniques.

Chapter 7: Describes a framework to expand short texts in a task specific way, such that using the expanded texts, the performance of the task (such as document organization, classification, clustering, and the like) improves. The framework is generic enough to be adapted by many ML NLP and IR tasks.

In Chapter 8 we conclude our research work and make closing remarks.

Chapter 2

Associating Categories with Documents*



*__Published__: Bairi, R. B., Ramakrishnan, G., & Sindhwani, V. (2014, July). Personalized classifiers: evolving a classifier from a large reference knowledge graph. In Proceedings of the 18th International Database Engineering & Applications Symposium (pp. 132-141). ACM.

Chapter Summary

When faced with the task of organizing and managing a large digital library of documents through an automatic text categorization system, the immediate challenge is that of the choice of categories itself. We expect that categories would be collectively representative of the documents in the digital library. A completely automated approach to category creation from the underlying collection could be prone to myopic or noisy category creation. On the other hand, a completely manual approach to the creation of categories could be cumbersome and expensive. Through this work, we propose an intermediate solution, by which a global, collaboratively developed Knowledge Graph of categories can be adapted to a local document categorization problem effectively. We present a principled approach in order to develop an organization-specific, multi-class, multi-label document classification system from scratch. We model our classification problem as one of inferring structured labels in an Associative Markov Network (AMN), in which the label space is derived from a global category catalog. By incorporating the outputs of various classification techniques (such as SVMs, Topic Models, Knowledge Mining, and the like) as dynamic features in AMN along with the document similarity measures (such as Jaccard, Cosine, BM25, and the like) as static features, we present a novel framework for learning to associate categories with documents. We evaluate our approach using document collections from Reuters RCV1-v2 and arXiv with Wikipedia as a global catalog and show that our techniques are effective and practical.

2.1 Introduction

With the growth of digital data in the form of news, blogs, web pages, scientific articles, books, images, sound, video, social networks and so on, the need for effective categorization systems to organize, search and extract information becomes self-evident and imperative.

A natural question in building a categorization system is "what should the representative categories be?" Imagine a digital library in an academy (or an organization) with thousands of articles on various topics. If a librarian wants to set up a good categorization system, he or she has to first come up with a set of categories that can effectively represent the documents in the library. Categories that are generic such as News, Entertainment, Technical, Politics, Sports, and the like might not help. Each of such categories accumulates thousands of articles and searching for the required piece of information continues to be challenging. On the other hand, fine grained category creation needs domain experts and is a laborious task.

The next option for the librarian is to look for some predefined category systems. For example, categories from Reuters RCV1-v2 Text Categorization Test Collection[1], Yahoo Directories[2] or DMOZ[3] can be adopted and one could hope to fit the existing documents in the library into these categories. How good is such an adoption? For example, Wikipedia Miner[4] [95] assigns categories to a document on the basis of (a) a similarity between the document content and the category description and (b) the coherence between category. We conducted an experiment on the Reuters-21578 data set. For a sample of the Reuters documents, we identified categories for each using Wikipedia-Miner and tabulated the results in Table 2.1. Interestingly, in some cases (Doc ID 5036 and 5038)the categories of Reuters and Wikipedia-Miner are different form each other. In other cases, Wikipedia-Miner categories are very fine grained. This indicates that adoption of predefined categories may not work well in all cases.

[1] http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[2] http://dir.yahoo.com/

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

[4] https://github.com/dnmilne/wikipediaminer/wiki

Doc ID	Reuters Categories	WikipediaMiner Categories
5038	australia, usa	Jiffy Lube, Automotive industry, Motor oil
5037	soybean, oilseed, corn, grain, ussr, usa	Soybean, Maize, United States Department of Agriculture, Wheat, Agriculture
5036	usa	Hepatitis B, Hepatitis, Hepatitis B vaccine, Vaccine, Merck & Co.
5027	money-supply, usa	Money supply, Moving average, Federal Reserve System, Money, Economist

Table 2.1: Categories from Reuters versus categories from Wikipedia

There is another practical issue with a predefined category system. Even though the category system contains most of the categories that can cover the documents in the digital library (known as *technical coverage*), it may not have *intent coverage*. In other words, the librarian may not want fine grained categories in some areas and may want it in some other areas. For example, an institute that works in the filed of computer science may not wish to have fine grained categories of *Bacteria's* or *Genes* even though there are documents about classification algorithms for bacteria or gene classification in its digital library. It may simply require to categorize all such documents under a single category termed *Bio-Informative*. However, it may wish to have fine grained categories in the Machine Learning area such as *Classification*, *Clustering*, *Active Learning*, *Kernel Learning*, *and the like*. The opposite case may be true for an institute that works in the filed of Bio Technology.

Another practical challenge in a predefined or fixed category system is maintaining *temporal relevance* of the category system. The categories that can accommodate the documents in a digital library today may need more categories in future when new documents are added to the library. A good categorization system should detect the emergence of these new categories and evolve its classifier to accommodate the new categories. The primary reasons for failing to achieve *temporal relevance* are the assuming of fixed categories and failing to update the categories to reflect the evolution of new concepts in a growing

document collection.

To summarize, adopting a predefined category catalog from an existing classification system (such as the Reuters text classification dataset) may not be always suitable. Such a strategy could lead to (i) under or over specific categories (ii) failure to capture user intention: and (iii) failure to evolve with time.

All these practical issues in designing a document categorization system lead us to conceptualize an alternative strategy that could solve these problems. We propose to define an evolving global catalog of possible categories where each category is accompanied by some description of that category. It is not unreasonable to construct such catalogs. In fact, we have already collaboratively built knowledge bases such as Wikipedia which can function as a global catalog of categories. Wikipedia's five million articles cover the terminology required of nearly any document collection [95], which could render it a good candidate for Global Category Catalog (GCC). Next we need sound techniques to adopt this global catalog to our local collection of documents. In this chapter, we propose a technique to solve this problem by learning a model that would project the documents into a localized subset of the global categories by capturing various signals from the documents, categories and possibly the structural knowledge in the global category catalog. In Figure 2.1 we pictorially depict the principle that forms the basis of our categorization system. We compare our system against existing algorithms and baselines and show that our approach is effective and practical.

2.1.1 Background

Text classification/categorization is an area in machine learning which is studied well under a variety of settings such as supervised, unsupervised,semi-supervised, and active learning.

Supervised text categorization entails learning from "training" data, in which predefined category labels are manually assigned to a set of documents. Various classifiers that are under headings such as "parametric", "non-parametric", "generative", "discriminate", and the like have been extensively studied by many researchers. Multi-label text classification

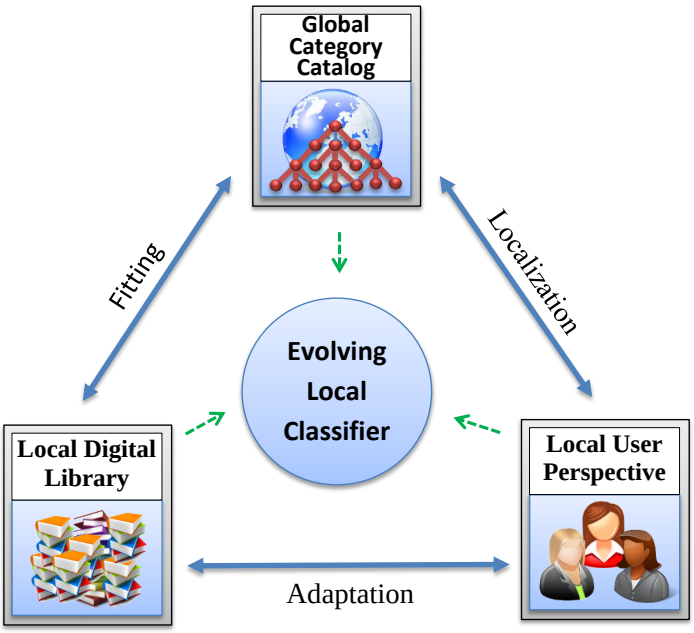


Figure 2.1: Categorization Triangle: Global Category Catalog may over-fit the local documents; Localization of the global catalog is needed to adapt the user perspective of categorization of documents in to the digital library.

systems allow the assignment of multiple category labels to a new document.For the construction of classification models, several multi-label learning algorithms [59, 144, 69] have been designed so that the correlation between classes can be exploited. The supervised learning techniques make use of labeled examples to learn the model parameters. The only input to our approach is the knowledge graph we present methods to evolving the model on the basis of feedback from the user.

Rousu et al. [118] present a hierarchical classification system that has a predefined class hierarchy. Their classification model is a variant of the Maximum Margin Markov Network framework, in which the classification hierarchy is represented as a Markov tree.

Text categorization systems that adopt unsupervised learning, identify topics or labels without requiring manual labeling of the data. A significant amount of work on clustering of texts [165, 130, 148] exists and text has found practical applications such as in the clustering of search results (http://clusty.com/). Approaches to unsupervised learning, such as LDA[17] CTM[16], PAM[85], NMF[6] identify topics as a group of prominent words. The topic of each cluster usually remains implicit in these approaches, though it would of course be possible to apply any keyword extraction algorithm to the resulting

clusters in order to find the characteristic terms. The determining of the right number of clusters, as well as the discovering of several hundred topics by using these techniques is challenging. In addition, finding a good representative and a grammatically correct topic name for a group requires additional effort.

Semi-supervised learning [32] uses a large amount of unlabeled data, along with labeled data to build classifiers. Since semi-supervised learning requires less human effort and is more accurate in some cases, it is of great interest both in theory and in practice. Many algorithms that make use of unlabeled data have been proposed [124, 32, 163]. Transitive learning, which is a special case of semi-supervised learning uses unlabeled data during training and classifies the unlabeled data using the learned model; it does not have any in-built mechanisms to classify unseen data. In some sense, our text categorization algorithm also makes use of unlabeled data, but it does not follow the semi-supervised setting. Moreover, we augment our semi-supervised learning approach with active learning (discussed in detail in Chapter 3) and update our model based on user feedback.

2.2 Formal Problem Statement and Solution Proposal

We now formally define our problem and the approach we developed to solve it.

2.2.1 Problem Definition

We assume that a global catalog of categories (GCC[5]) $C = \{C_i\}_{i=1}^{i=f}$ has been built with some description for each category. We further assume that an organization receives documents in batches $D_1, D_2, ...$, which, we believe, is a fair assumption. For example, organizations receive weekly,/monthly periodicals/or /magazines, an academic institute library receives reports or these at the end of the semester/year, and the like. The organization needs to adopt the global category catalog and logically build an organization specific category catalog $C^{org} \subseteq C$ and at the same time, evolve some models to classify all $d_i \in D_j$ into C^{org}. We assume the following major sub tasks specifically:

[5]For example, categories and article titles from the Wikipedia can form a GCC.

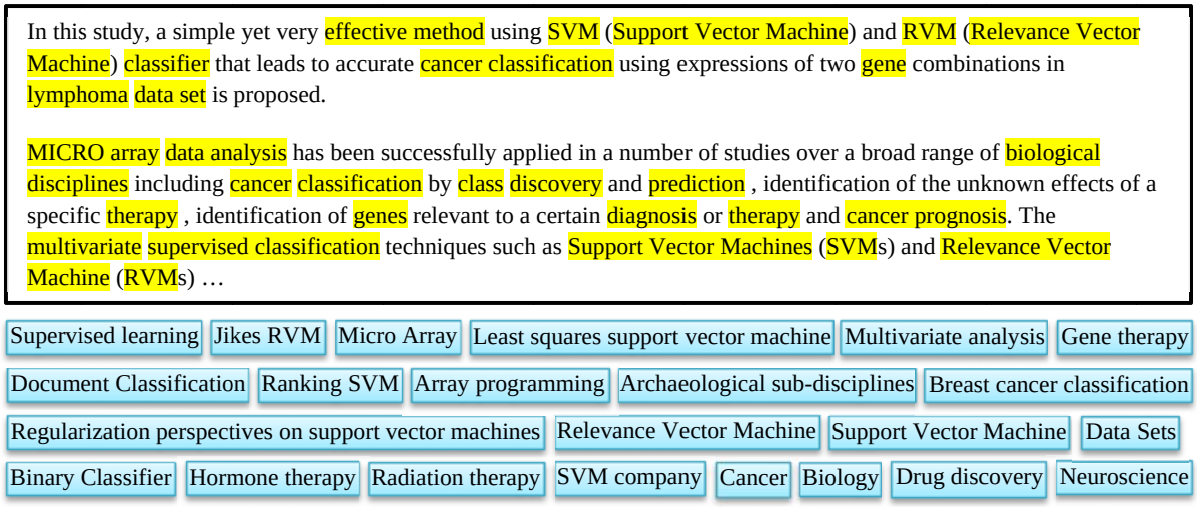


Figure 2.2: Document with detected keywords and candidate categories (only few shown)

1. Given batches of documents $D_1, D_2, ...$, identify $C^{org} \subseteq C$ through a combination of feature design and user feedback.

2. Evolve a multi-label, multi-class document categorization model to categorize documents into C^{org}.

The eventual goal is to accurately identify suitable categories $\{C_1, ...C_{T_i}\}$ for every input document $d_i \in D_j \ \forall i, j$.

If one could learn an SVM classifier for every category in the GCC, identifying all suitable categories for a document would entails the identifying of the classifiers label that the document as positive. However, learning these classifiers is prohibitively expensive because the GCC is usually very large (*for example* Wikipedia has five million titles) making it impractical to learn a model (like SVM) for every category in the GCC upfront. Hence it is a challenging task to develop a classification system that can subset the millions of categories that suit an organization. We attempt to solve this problem using a suite of active learning and knowledge propagation techniques that we explain next.

2.2.2 Our Approach

It has been observed that a document that is tagged with a category is expected to contain features such as keywords or/phrases that are indicative of that category [95].

For example, the text shown in Figure 2.2, contains several words or/phrases that are indicative of some of the category titles in the GCC (Wikipedia in our examples). We refer to such categories as *candidate categories*. However, some of these features could be either (a) misleading or (b) not relevant in determining the "interesting categories". As an illustration of (a), consider in Figure 2.2, the category "Jikes RVM", which means Java JVM and is not relevant to the document. Thus, the word "RVM" is misleading as a feature. On the other hand, though the category "Cancer" is relevant to the document, the user may want to restrict the choice of categories to the computer science domain, and may therefore not be interested in categories such as "Cancer', thus creating a case for (b). Our goal is to develop a personalized categorization system that has the capacity to evolve and learn the accurate identification of only relevant categories. This can be achieved by incrementally learning a classifier for each class, on the basis of user feedback. We expect the classifier training to result in feature weights such that the effect of the misleading and irrelevant features that are described above is minimized. This will eventually result in as many classifiers as the cardinality of C^{org}, which is, in the worst case, the same as C. In order to avoid the overhead of invoking the classifier for each class from C^{org} on every input document, we apply some simple index-based filters that discard categories from C^{org} which have little or no relevance to the document. We refer to the resulting smaller subset of categories as the candidate categories and denote this set by $C^{cand} \subseteq C^{org}$.

It has been observed that the categories that are assigned to a document either exhibit semantic relations such as "is-a", and "association"[6] or tend to be frequently assigned together (that is,they tend to co-occur) in a particular instance of the classification exercise. For example, with the Reuters RCV1V2 dataset, we observe that all pairs of categories that co-occur even once in the training dataset, co-occur at least in 7% of the documents. In fact, the co-occurrence of all category pairs is significant at 95% significance level. In other instances of classified data such as DMOZ[7] or the Yahoo! categories[8], we make an additional observation that co-occurring categories exhibit semantic relations such as "is-", or "association". For example, the category "Linear Classifier" is related to categories like 'Kernel Methods in Classifiers', "Machine Learning", and the like are observed

[6] `http://marciazeng.slis.kent.edu/Z3919/44association.htm`

[7] The open directory project at `http://www.dmoz.org`

[8] `http://www.dir.yahoo.com`

Semantic Relation	Description	Example Categories
synonym	Categories that mean exactly or nearly the same.	SVM and Support Vector Machines; Optimization and Mathematical Optimization;
association	An association signifies the connection between two categories. It can be unidirectional when one category includes the other within its description, or bidirectional where both categories use each other in their descriptions.	SVM and Kernel Methods; Optimization and Constraint Programs;
is-a	*is-a* is a consumption relationship between categories, when one category A is a subclass of another category B.	Decision Trees and Classification Algorithms; Linked List and Data Structures;

Table 2.2: Co-occurring categories and the relations that they exhibit

to occurrence as labels for a document on "Classifiers". Another illustration, is that of the categories "Supervised Learning" and "Document Classification" which exhibit a large amount of overlap in their textual descriptions. Table 2.2 lists examples of co -occurring categories and the relations they exhibit according to the Wikipedia catalog.

To summarize, we identify two types of informative features to identify relevant categories for each document: (i) a feature that is a function of the document and a particular category, such as the category-specific classifier scoring function evaluated on a document and (ii) a feature that is a function of two categories, such as their co -occurrence frequency or the textual overlap between their descriptions. We find that the Markov Network (MN) [119], is a very natural manner of modeling these two types of features. Next, we provide a more detailed description of our modeling of this problem as a Markov Network.

For every input document d, we construct a MN of the candidate categories C^{cand}, such that each node represents a candidate category $C_i \in C$ and edges represent the asso-

ciation between the categories. Modeling inter-category relations through edges serves two important purposes in our approach: i) When a new organization starts categorizing documents, the classifier models are initially not tuned. The only information that is available to the categorization system is the category description. It is not practical to assume a that comprehensive description will be available for every category. In such cases, the relationships between the categories can help to propagate descriptions across categories through their neighbors. ii) As a part of the learning of the model parameters, the system solicits user feedback on some of the suggested categories for a document. Based on the feedback, the category specific model (classifiers such as SVM) is updated. The category relationships help in propagating the learning to the neighbors. This reduces the amount of feedback that is needed to learn model parameters. We will illustrate both these advantages in our experimental section.

Our aim is to learn to assign a binary label (0/1) for every category node C_i in the above MN. Label 1 indicates that the category C_i is valid for the document d and 0 indicates invalid. The collective assignment of labels for all the nodes in the Markov network produces relevant categories for the document d. As we will see later in the thesis, optimal assignment of these labels can be achieved through MAP inference using Integer Linear Programming.

Since the categories that are assigned to a document happen to be related with each other, we employ a Markov Network variant that is known as an AMN [139]. The subsequent sections explain our techniques of modeling the categorization problem in AMN with other classifiers (such as SVM) and similarity functions that act as node features in the AMN framework.

In algorithm 1, we present the steps of our overall category discovery process.

Figure 2.3 illustrates the overall process of evolving a personalized classifier. The Personalized Classifier component detects and assigns personalized categories for the input documents by using GCC, Constraints and other Model Parameters. The Active Learner component chooses selected documents and categories therein in order to solicit user feedback. The feedback is used to update the constraints and learn the personalization model parameters.

Algorithm 1 Batch Category Discovery Process

1: Input : Global Category Catalog ($C_1 ... C_f$), Document batch D_j, Previously learned Model parameters θ^t

2: Output : Categories relevant to each document $d_i \in D_j$, Updated Model parameters θ^{t+1}

3: **for all** $d_i \in \{D_j\}$ **do**

4: Retrieve categories from GCC and prune uninteresting categories for d_i

5: Build an Associative Markov Network (AMN) over categories from the global category catalog

6: Compute node potentials φ_i and edge potentials ψ_{ij} using the model parameters θ^t and the structural constructs

7: Perform 0/1 inference on the above AMN. Assign all categories labeled as 1 to document d_i

8: **end for**

9: Seek feedback on the Category assignment from the user (In Chapter 3 we present a joint *Active Learning* technique over the Document and Category space to select the most uncertain document-category pairs for seeking feedback and hence reduce the cognitive load on the users.)

10: Solicit user feedback F for selected documents and categories and update model parameters, using learner $\mathcal{L}$

$$\theta^{t+1} = \mathcal{L}\left(\theta^t, F\right)$$

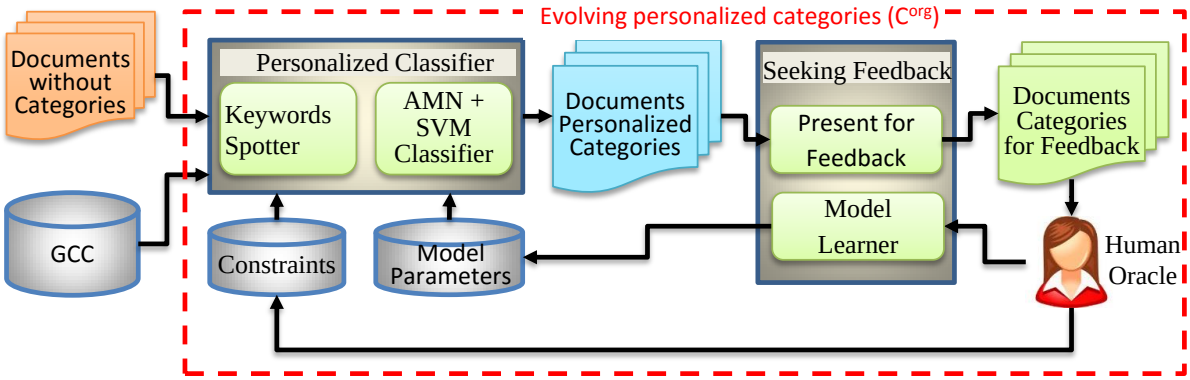


Figure 2.3: Architecture of evolving personalized categorization system.

The important notations that used in this chapter are listed below:

d or d_i	Single input document; $\mathbf{d}$ - document vector; $D_1, D_2, ...$ batches of input documents.
C_i	Category in GCC represented by the i^{th} node in the Markov Network (MN)
$\mathbf{x}_i$	Feature vector for i^{th} node (C_i) in the MN
x_i	i^{th} feature in the feature vector $\mathbf{x}_j$
$\mathbf{x}_{ij}$	Feature vector for the edge connecting i^{th} and j^{th} node in the MN
$\mathbf{x}$	Set of all node and edge feature vectors in the MN
y_i	Set of all node labels in the MN
$\mathbf{y}$	Set of all node labels in the MN
ϕ_i	Node potential of i^{th} node in the MN
φ_{ij}	Edge potential of the edge connecting i^{th} and j^{th} node in the MN

Table 2.3: Notations used in this chapter

2.3 Building the Personalized Classifier Model

We now describe the our technique for building a personalized document organization system. We start with the background on Markov Network, Associate Markov Networks, and then describe solving our problem of associate categories with the documents using these techniques.

2.3.1 Markov Random Field

A Markov Random Field, also known as Markov Network is an undirected graph with a set of cliques $\mathbb{C}$ and a *clique potential* c, which is a non-negative function that is associated with each $c \in \mathbb{C}$. In the context of classification, we consider conditional MRFs which define the distribution

$$P(\mathbf{y}|\mathbf{x}) = \frac{\prod_{c\in\mathbb{C}} \phi_c\left(\mathbf{x}_c, \mathbf{y}_c\right)}{\sum_{\mathbf{y}'} \prod_{c\in\mathbb{C}} \phi_c\left(\mathbf{x}_c, \mathbf{y'}_c\right)} \tag{2.1}$$

where $\mathbf{x}_c$ and $\mathbf{y}_c$ are the features and labels of nodes in the clique c. Here, the potential ϕ_c is a mapping from features and labels to a positive value. The higher the value, the more likely it is that the labels $\mathbf{y}_c$ are correct for the features $\mathbf{x}_c$. The denominator in equation 2.1 is called the partition function,(usually denoted by Z,) and is essentially a sum of all possible labeling s.

2.3.2 Associative Markov Network

To simplify the problem, the size of the cliques is usually restricted to either one or two. This results in a pairwise MRF,in which only the node and edge potentials are considered. For a pairwise MRF with the set of edges E, equation 2.1 simplifies to

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod \varphi\left(\mathbf{x}_i, y_i\right) \prod \psi\left(\mathbf{x}_{ij}, y_i, y_j\right) \tag{2.2}$$

Note, here the node features $\mathbf{x}_i$ are computed by considering the node description and the input document text. Hence the distribution mentioned above is for a given document d.

Again, Z denotes the partition function given by $Z = \sum_{\mathbf{y}'} \prod \varphi\left(\mathbf{x}_i, y'_i\right) \prod \psi\left(\mathbf{x}_{ij}, y'_i, y'_j\right)$. Note that in equation 2.2, there is a distinction between the node features $\mathbf{x}_i \in \mathbb{R}^{d_n}$ and the edge features $\mathbf{x}_{ij} \in \mathbb{R}^{d_e}$.

2.3.3 Building the AMN Model from Categories

For a given document d, we create a MN $G = (N, E)$,the nodes N are the candidate categories from the global category catalog. Edges are added on the basis of the relationships between the categories. The relationship can be inferred from techniques as simple

as capturing the jaccard/cosine similarity between the descriptions of categories C_i and C_j or it can be more complex semantic relationships. We assume that there exists good measures to identify relationships between categories.

In an AMN, only node and edge potentials are considered. For an AMN that has a set of nodes N and edges E, the conditional probability of label assignment to nodes is given by 2.2. We use notation $\mathbf{x}_i$ to denote a set of node features for C_i and $\mathbf{x}_{ij}$ to denote the set of edge features for the edge that connects C_i and C_j. y_i and y_j are the binary labels for nodes C_i and C_j.

Note, here the node features $\mathbf{x}_i$ are computed by considering the node description and the input document text. Hence the distribution mentioned above is for a given document d.

The use of the log-linear model is a simple method to define the potentials φ and ψ . In this model, a weight vector w_k is introduced for each class label $k = 1..K$. The node potential φ is then defined as $log\varphi\left(\mathbf{x}_i, y_i\right) = \mathbf{w}_n^k \cdot \mathbf{x}_i$ where $k = y_i$. Accordingly, the edge potentials are defined as $log\psi\left(\mathbf{x}_{ij}, y_i, y_j\right) = \mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}$ where $k = y_i$ and $l = y_j$. Note that there are different weight vectors $\mathbf{w}_n^k \in \mathbb{R}^{d_n}$ and $\mathbf{w}_e^{k,l} \in \mathbb{R}^{d_e}$ for the nodes and edges.

By using the indicator variables y_i^k we can express the potentials as

$$log\varphi\left(\mathbf{x}_i, y_i\right) = \sum_{k=1}^{K} \left(\mathbf{w}_n^k \cdot \mathbf{x}_i\right) y_i^k \tag{2.3}$$

$$log\psi\left(\mathbf{x}_{i,j}, y_i, y_j\right) = \sum_{k=1}^{K} \left(\mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}\right) y_i^k y_j^l \tag{2.4}$$

where y_i^k is an indicator variable which is 1 if node p_i has label k and 0, otherwise.

To bring in the notion of association, we introduce the constraints $\mathbf{w}_e^{k,l} = 0$ for $k \neq l$ and $\mathbf{w}_e^{k,k} \geq= 0$. This results in $\psi\left(\mathbf{x}_{ij}, k, l\right) = 1$ for $k \neq l$ and $\psi\left(\mathbf{x}_{ij}, k, k\right) \geq 1$. The idea here is that the edges between the nodes with different labels should be penalized over edges between equally labeled nodes.

The objective is to maximize $P\left(\mathbf{y}|\mathbf{x}\right)$, which is equivalent to maximizing $log\, P\left(\mathbf{y}|\mathbf{x}\right)$. By substituting equations 2.3 and 2.4, the objective becomes

$$\max_{\mathbf{y}} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\mathbf{w}_n^k \cdot \mathbf{x}_i\right) y_i^k + \sum_{(ij)\in E} \sum_{k=1}^{K} \left(\mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij}\right) y_i^k y_j^k - \log Z\left(x\right) \tag{2.5}$$

Note that the partition function Z only depends on $\mathbf{w}$ and $\mathbf{x}$, but not on the labels $\mathbf{y}$, hence need not be computed while solving the above optimization problem.

2.3.4 Inferring Categories for a Document

The problem of inference is the determining of the most relevant categories for a new input document. That is, a subset of nodes (that is, categories) from G which has the highest probability of being relevant to the input document, has to be selected. To model this selection, we attach a binary label $\{0, 1\}$ to a node. A node C_i with label 1 is considered to be a valid category for the input document it is considered to be invalid if its label is 0. Node and edge potentials are calculated with K=2, as shown in equations (2.3) and (2.4).

Correctly determining the categories for the input document is equivalent to solving the optimization problem in (2.6).

$$\max_{\mathbf{y}} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(w_n^k \cdot x_i\right) y_i^k + \sum_{(ij)\in E} \sum_{k=1}^{K} \left(w_e^k \cdot x_{ij}\right) y_{ij}^k \tag{2.6}$$

$$s.t. \quad y_i^k \geq 0, \quad \forall i, k;$$

$$\sum_{k=1}^{K} y_i^k = 1, \ \ \forall i$$

$$y_{ij}^k \leq y_i^k, \ \ y_{ij}^k \leq y_j^k, \ \ \forall ij \in E, k$$

$$y_i^0 \ = 1 \ \ \forall i \text{ with Filter Constraints}$$

The variables y_{ij}^k represent the labels of two nodes that are connected by an edge. The last two inequality conditions are a linearization of the constraint $y_{ij}^k = y_i^k \wedge y_j^k$;

Filter Constraints, which are explained in section 2.5, are the user imposed constraints which force a category not to be selected for a document. Adding *filter constraints* rather than filtering out the categories before constructing the AMN itself, allows the propagation of the influence to the neighboring nodes. For example, if category A is strongly related to category B (that is, the edge potential between A and B is high), adding a filter constraint on category A, also forces category B to not be selected for the document during the

inference. This happens naturally in AMN because, the MAP inference attains maximum value when the strongly associated neighboring nodes get similar labels. This is one of the main reasons for modeling the category identification problem using the AMN framework.

If we have a budget T on the number of categories to be chosen for a document, we can add the constraint $\sum_{i=1}^{N}(y_i^1) \leq T$. Then, (2.6) yields the top T categories for the input document. However, we strongly recommend not to apply a the budget constraint at this stage. This is because, due to the associative nature of AMN, if categories A and B are strongly related and the category A is selected for a document, then it is highly likely that category B will also be selected. Hence, AMN does not enforce diversity in the selection of the category. Due to this, selecting only those categories that are relevant to the top T is likely to include very similar categories. We recommend applying the diversity or summarization techniques that are discussed in Chapters 4 and 5 respectively in the selection of T diverse categories at the document or the collection level.

2.3.5 Learning Feature Weights

We now discuss the formulation of objective function for learning feature weights and then present a technique for optimizing the objective function.

2.3.5.1 Objective Function

The standard approach to learning the weights $\mathbf{w}$ given N training examples $(\mathbf{x}, \widehat{\mathbf{y}})$ is to maximize the $\log(\widehat{\mathbf{y}}|\mathbf{x})$, with an additional regularization term.Here $\widehat{\mathbf{y}}$ denote the true node labels. A method that is proposed by Taskar et al. [139] is the maximizing of the margin of confidence in the true label assignment $\widehat{\mathbf{y}}$ over any other assignment $\mathbf{y} \neq \widehat{\mathbf{y}}$. They show that the margin-maximization criterion provides significant improvements in accuracy over a range of problems. In this approach, the quadratic program (QP) takes the following form:

$$\begin{aligned} &\underset{\mathbf{w},\xi}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{w}\|^2 + c\xi \\ &\quad s.t.\ \mathbf{w}\mathbf{X}\widehat{\mathbf{y}} + \xi \geq \max_{\mathbf{y}} \mathbf{w}\mathbf{X}\mathbf{y} + (N - \mathbf{y}.\widehat{\mathbf{y}}_{\mathbf{n}})\,; \ \ w_e \geq 0 \end{aligned} \tag{2.7}$$

Algorithm 2 Cutting Plane Algorithm for Learning Feature Weights

1: Input : Training set $(\mathbf{x}, \widehat{\mathbf{y}})$ with N examples

2: Output : Feature weights $\mathbf{w}$

3: Randomly initialize $\mathbf{w}^{(0)}$ and $\xi^{(0)}$

4: $\mathcal{K} = \{\}$

5: **while** Improvement in the objective value in Eq 2.7 is significant **do**

6: $\kappa^{(t)} = \max_{\mathbf{y}} \mathbf{wXy} + (N - \mathbf{y}.\widehat{\mathbf{y}}_{\mathbf{n}})$

7: $\mathcal{K} = \mathcal{K} \cup \left\{\kappa^{(t)}\right\}$

8: Compute $\mathbf{w}^{(t+1)}$ by solving the following QP

$$\underset{\mathbf{w},\xi}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2 + c\xi$$

$$s.t.\ \mathbf{wX}\widehat{\mathbf{y}} + \xi \geq \kappa_i \quad \forall \kappa_i \in \mathcal{K}$$

$$\mathbf{w}_e \geq 0$$

9: **end while**

2.3.5.2 Cutting Plane Algorithm for Learning Feature Weights

Taskar et al.[139], present a technique for solving 2.7 using its dual form which has a variable for each normalization constraint in Equation 2.6 and two variables for each of the inequality constraints. By substituting the dual variables into the constraints of the primal form, a QP is formed and solved for $\mathbf{w}$. With a large AMN graph (with several thousand nodes and tens of thousand edges, solving this QP becomes challenging using the QP solvers that are available in the public domain. Here we present a cutting plane style algorithm [74] to directly solve the QP Equation 2.7 in the primal form, in Algorithm 2.

The algorithm starts with a random $\mathbf{w}$ and computes the loss augmented inference (LAI) value $\kappa^{(t)}$ for the current weights $\mathbf{w}^{(t)}$. It maintains all the LAI values seen until the current step t, that is $\mathcal{K} = \left\{\kappa^{(0)}...\kappa^{(t)}\right\}$. A new value of $\mathbf{w}^{(t+1)}$ is computed by applying the constraints over all the values of $\mathcal{K} = \left\{\kappa^{(0)}...\kappa^{(t)}\right\}$. This is because a weight vector

$\mathbf{w}^{(t+1)}$ that fits the training examples better than the previous weight vectors $\left(\mathbf{w}^{(0)}...\mathbf{w}^{(t)}\right)$ will have $\mathbf{w}^{(t+1)}\mathbf{X}\widehat{\mathbf{y}} + \xi$ values that are at least as large as the previous LAI values $\mathcal{K} = \left\{\kappa^{(0)}...\kappa^{(t)}\right\}$. The constraint set grows in each iteration, improving the weight vector. The iteration stops when the improvement in the objective value is less than the threshold.

2.3.5.3 Handling Unbalanced Classes

Our training data is highly skewed. Usually a document will have only a few relevant categories. Hence, most of the category nodes in AMN will have the label 0 and only a few will have the label 1. *For example*, in the Reuters RCV1-v2 collection we have 642 categories, whereas, most of the documents belong to less than 15 categories. Hence the AMNs that are constructed for such documents will have 642 nodes, of which less than 15 nodes will have label 1 and the rest will have label 0. In order to handle this skew in the training data, we explored two strategies, namely, (i) penalty variation and (ii) node boosting.

Penalty Variation We introduce two hyper parameters C_p and C_n that award different penalties for a label 1 mismatch and a label 0 mismatch in the max-margin training objective:

$$\begin{aligned}
&\min \frac{1}{2}\left\|\mathbf{w}\right\|^2 + c\xi \\
&s.t.\ \mathbf{w}\mathbf{X}\widehat{\mathbf{y}} + \xi \geq \max_{\mathbf{y}} \mathbf{w}\mathbf{X}\mathbf{y} + C_p\left(N^1 - \mathbf{y^1}.\widehat{\mathbf{y}}_\mathbf{n}^\mathbf{1}\right) \\
&\quad + C_n\left(N^0 - \mathbf{y^0}.\widehat{\mathbf{y}}_\mathbf{n}^\mathbf{0}\right);\ \ w_e \geq 0
\end{aligned}$$

where N^1 and N^0 are the total number of 1s and 0s in the training examples; $\mathbf{y^1}.\widehat{\mathbf{y}}_\mathbf{n}^\mathbf{1}$ are the total number of disagreements between the actual label 1s and the inferred label 1s. Similarly $\mathbf{y^0}.\widehat{\mathbf{y}}_\mathbf{n}^\mathbf{0}$ are the disagreements between the actual label 0s and the inferred label 0s. Empirically we have observed that training with this technique is highly sensitive to the hyper parameters C_p and C_n. Setting $C_n = 1$ and $C_p = (\#\text{ nodes with label 0}) / (\#\text{ nodes with label 1})$ generally yields better results.

Node Boosting In this technique, we replicate all nodes that are labeled 1 (and edges that are labeled 11) multiple times in the training examples, so that the total number of nodes that are labeled 1s and 0s is almost equal. We use this modified AMN for training as well as for inference. Empirically we have found this technique to be far more useful than the previous one.

2.3.6 Personalization of Classification

Personalization is the process of learning to categorize by using categories that are of interest to an organization. As discussed in previous sections, the AMN classifier suggests categories by virtue of solving the inference problem in (2.6). When the AMN classifier is initially deployed, the categories that are suggested depend completely on the static features. It is quite possible for the system to suggest categories that may not be acceptable to the user(s).The following are plausible reasons: for this i) The description for each category in the global catalog is not exhaustive. ii) the static node features are not discriminative across categories, leading to inter category confusion and iii) the user may not be interested in certain (classes of) categories.

Personalization is achieved by soliciting feedback from a human oracle about the system suggested categories and by using it to retrain the system parameters. The feedback is solicited as "correct", "incorrect" or "never again" for the categories that are assigned to a document by the system. In the next few sections we describe how this feedback is used to train our model for personalization.

2.4 Feature Engineering for Personalization

We now present discussion on creating node and edge features for the AMN.

2.4.1 Node Features

Node features in AMN determine the relevance of a category to the input document d. We divide the node features $\mathbf{x}_i$ into two types: i) dynamic node features $\mathbf{x}_i^d$ and ii) static node features $\mathbf{x}_i^s$. The node feature vector becomes $\mathbf{x}_i = \left[\mathbf{x}_i^s; \mathbf{x}_i^d\right]$.

2.4.1.1 Dynamic Node Features

The dynamic node features aid in the personalization of GCC. These feature values are computed using the category specific classification model that is trained from the user feedback that is available for the documents that have been examined thus far. Essentially, we learn an SVM model for every category on the basis of our active learning and user feedback setting which we explain in detail in Chapter 3. It is possible to consider the use of other machine learning models such as decision trees, logistic regression, and the like. Our choice of SVM was based partly on the fact that all of our baselines employ SVMs and partly on the fact that SVMs are known to yield high accuracies. We employ the decision function of the classifier as a node feature in the AMN. The SVM decision function for each category node takes the document d (as a TF-IDF vector),as input and evaluates $P_{C_i}(d) = \mathbf{w}_{C_i}^T \mathbf{d} + b_{C_i}$, where $\mathbf{w}_{C_i}$ and b_{C_i} are the SVM parameters that are learned for the category C_i. The output of the SVM decision function is positive if C_i is relevant to the document d and negative if not relevant. We also treat the output of the decision function as being 0 if the SVM model is not available for the category C_i. We introduce two features in the node feature vector $\mathbf{x}_i$ namely, SVM^1 and SVM^0, which are denoted using the notation SVM_i^1 and SVM_i^0.

The feature value is computed as follows:

$$SVM_i^1 = \begin{cases} \gamma_i P_{C_i}(d) & if \;\; P_{C_i}(d) \geq 0 \\ 0 & Otherwise \end{cases}$$

$$SVM_i^0 = \begin{cases} -\gamma_i P_{C_i}(d) & if \;\; P_{C_i}(d) < 0 \\ 0 & Otherwise \end{cases}$$

γ_i is the *damping factor*, which reflects the confidence in the SVM classifier for the category C_i. We believe that when the classification system is initially deployed, the SVM models are not sufficiently trained. Hence, the SVM feature values might not initially provide reliable signals in deciding the relevance of a category to the input document. As the categorization system matures by active learning and user feedback, the SVM models get trained such that we can begin to trust the SVM score progressively. We can control this

through the *damping factor* γ_i, which increases for a category along with the amount of feedback that is received for that category. We define γ_i to be:

$$\gamma_i = \begin{cases} 0 & \text{if } confidence\,(P_{C_i}) < \mathcal{T} \\ confidence\,(P_{C_i}) & \text{otherwise} \end{cases}$$

where $\mathcal{T}$ is the user defined threshold, and $confidence\,(P_{C_i})$ is the $\xi\alpha - estimator$ of the F1 score of P_{C_i} which is computed as proposed by Joachims [70]. These estimators are developed on the basis of the idea of the leave-one-out estimation of error the rate. However, leave-one-out or cross validation estimation is a computationally intensive process. On the other hand, $\xi\alpha - estimator$ [70] can be computed at no extra cost immediately after training every Svm_{C_i} by using the slack variables ξ_i in the SVM primal objective and α_i Lagrange variables in the SVM dual formulation.

Due to the associative property of AMN, the SVM parameters that are learned for a node can also influence the label assignments of its neighbors. In other words, if there is a strong edge potential between categories C_i and C_j, the SVM score propagates from C_i to C_j. This helps in correct detection of the label of node C_j even though there may not be a trained SVM classifier available for node C_j. The example in Figure 2.4 illustrates the knowledge propagation between closely associated (that is, with high edge potential) nodes. This is precisely that which we want to model using an AMN.

2.4.1.2 Static Node Features

The values of static features do not change as the system evolves. Global features aid in capturing the structural similarity through the combination of different kernels. In the absence of local features, global features help in determining the similarity between a node and the input document. This situation arises when the system is initially deployed in an organization and has not learned any organization specific model parameters. We considered standard kernels such as Bag of Words kernels, N-gram kernels and Relational kernels in our setup.

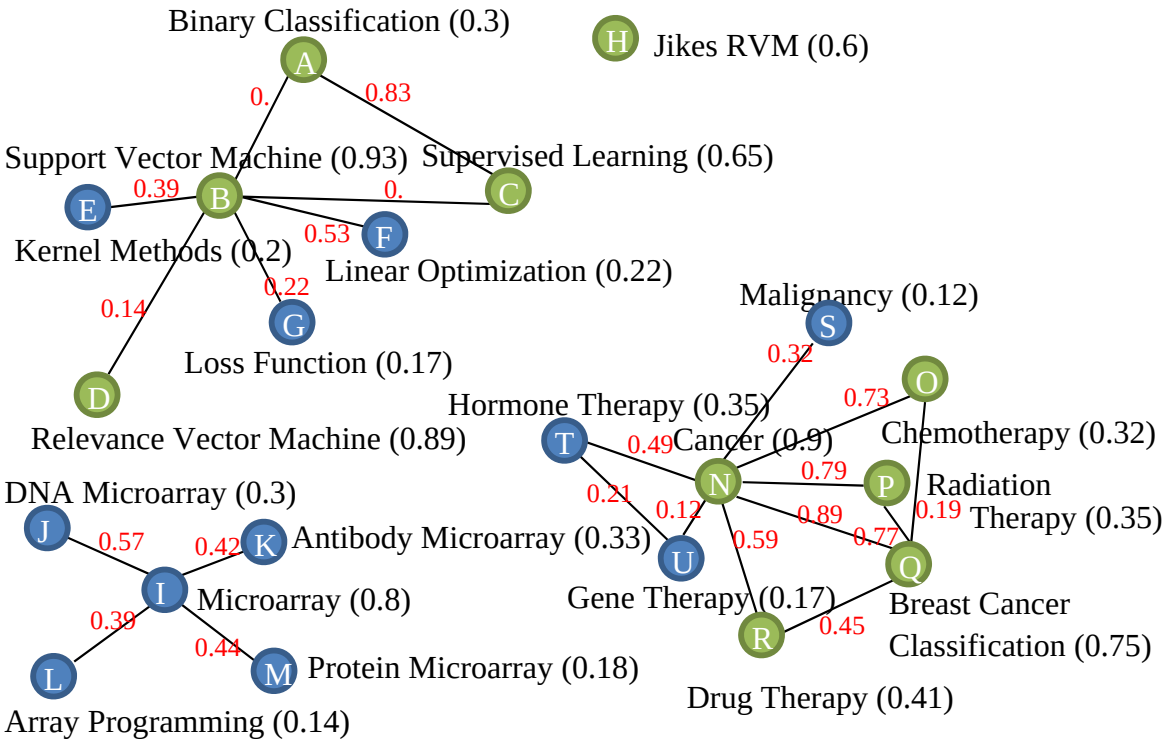


Figure 2.4: Knowledge Propagation: 1. Nodes B and C with label 1 force the strongly associated neighbor node A to assume label 1. The knowledge from node B and C propagates to node A. 2. Though node I seems to be valid for the document (with high node potential), given the context, it is not. Strongly associated neighbors of I, that is, nodes J, K, L, and M which have low node potentials force the node I to attain label 0. Again knowledge flows from J, K, L, and M to I.

2.4.2 Edge Features for Knowledge Propagation

Edges between categories in a Markov Network represent some kind of association between them. An Edge feature vector ($\mathbf{x}_{ij}$) contains feature values that encourage the categories C_i and $C_j$which are connected by the edge to have the same label if there is a strong relationship between them. The strength of the relationship is discovered through combinations of multiple Kernels. Let $K_1(C_i, C_j) \cdots K_M(C_i, C_j)$ be M Kernels that measure the similarity between C_i and C_j. Example Kernels include Bag-of-Words Kernel, Bi-gram Kernel, Trigram Kernel, Relational Kernel, and the like. Further we assume (without loss of generality) that these Kernels return normalized values (between 0 and 1). We define the feature vector $\mathbf{x}_{ij}$ as having M features that signal $y_i = 1$, $y_j = 1$ and M features that signal $y_i = 0$, $y_j = 0$. The feature vector $\mathbf{x}_{ij}$ is defined as follows $\forall 1 \leq m \leq M$

$$\mathbf{x}_{ij}[m] = K_m(C_i, C_j) \times (\log \varphi(\mathbf{x}_i, 1) + \log \varphi(\mathbf{x}_j, 1))$$

$$\mathbf{x}_{ij}[M + m] = K_m(C_i, C_j) \times (\log \varphi(\mathbf{x}_i, 0) + \log \varphi(\mathbf{x}_j, 0))$$

Note that $\log \varphi(\mathbf{x}_i, 1)$ is the node potential of C_i when it is labeled 1 and $\log \varphi(\mathbf{x}_i, 0)$ is the node potential when it is labeled 0. Essentially, when the similarity between nodes C_i and C_j is high, these features collectively favor label 1 on the nodes C_i and C_j, and label 0 otherwise.

2.5 Personalization from Category Constraints

2.5.1 Filter Constraints

In the process of personalizing the global catalog, users can indicate (through feedback) that a category C_i when suggested by the system should never reappear in the future categorization, because the organization is not interested in that category. For *for example* an organization working in the core area of computer science may not be interested in detailed categorization of cancers, even though there may be some documents that contain the classification algorithms for different types of cancers. The system remembers this feedback as a *filter constraint*. By *filter constraint* for a category C_i, we mean the inference

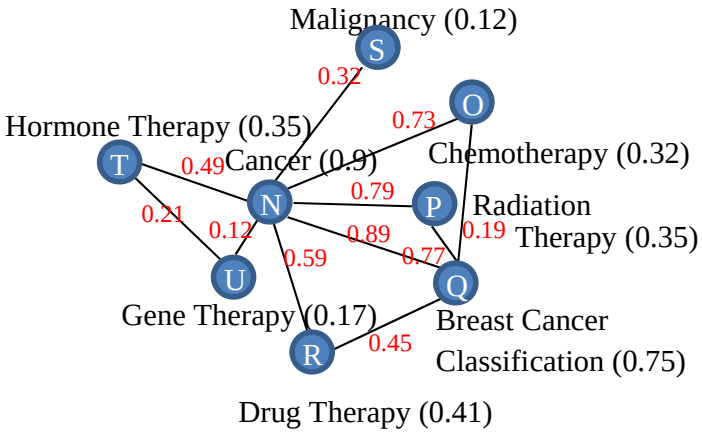


Figure 2.5: Constraint Propagation: By applying the "never again" constraint on node N, the label of Node n is forced to 0. This forces labels of strongly associated neighbors (O,P,Q,R) to 0. This is due to the AMN MAP inference, which attains maximum when the labels of these neighbors (with high edge potentials) are assigned label 0.

that is that is subject to a constraint set that includes $y_i^0 = 1$ as in Equation 2.6. If categories C_i and C_j are *related*, we would expect the effect of this constraint to propagate from C_i to C_j and encourage y_j^0 to also become 1. As shown in the example in Figure 2.5, if the user suppresses the category *Cancer* by introducing a hard constraint, the AMN inference will try to suppress the *related* categories as well. This is precisely what we want to model using an AMN.

2.5.2 Bulk Constraints

The key activity in the evolution of the classifier is timely feedback by the users,which enables the training of the the node specific classifiers in the AMN. This involves the inspecting of the categories that are assigned by the system to the documents and providing of the feedback “Correct”, “Incorrect” or “Never again”. This can become a laborious task when the number of categories that needs to receive feedback increases. In order to reduce the cognitive load in providing the feedback, we developed a few tools along with the Graphical User Interface (GUI).

These tools help the user to filter the assigned categories on the basis of the node potential threshold, and the descendants/ancestors of a particular category. Multiple filter criteria

can be combined using "AND", "OR", and "NOT" logical operators. Users can then apply feedback to all the filtered category at once in a bulk fashion. This reduces the cognitive load on the users.

The tool also facilitates the users to get all the neighbors of a category node and exercise bulk feedback on them. This is useful when we require to filter out an out of domain categories. For instance, in the example in Figure 2.4, if the user wants to apply "Never again" constraints to the category "Cancer" and all its related categories, it can be done all at once by choosing all the neighbors. In addition, users can also select neighbors that have a strong association with the category by thresholding on the edge potentials. This is useful when applying constraints to the strongly associated neighbor categories of a category.

2.6 Candidate Category Selection

Thus far we had conveniently disregarded the problem of spotting important keywords or phrases from the text of input documents that can map to candidate categories. However, this is an important phase in the identification of the the right choice of candidate categories. We refer to this phase as the *candidate selection* phase, and further identify two stages within this phase: (i) keyphrasing : identifying important words and phrases from the input document and (ii) candidate detection : selecting a set of categories from GCC that relate to these phrases or words.

2.6.1 Keyphrasing

This stage identifies words or phrases that contribute significantly to identifying the document's categories. We consider nouns (and sequence of nouns) to be more important in detecting a category. After POS tagging the input document, we select the phrases as a sequence of words that have noun (and adjectives if any)as a POS tag. Once the phrases are detected, we compute the importance of a phrase p as

$$Importance\,(p) = count\,(d_p) \times \log\left(\frac{|C|}{1 + \log\,(count\,(C_p))}\right)$$

where $count\,(d_p)$ = number of occurrences of the phrase p in input the document, $count\,(C_p)$ = the number of category titles the contain the phrase p and $|C|$ =the total number of categories in GCC. All phrases with importance greater than a threshold are considered as key phrases.

2.6.2 Candidate Detection

Key phrases that are detected in the previous step are used to look up an index of category titles and descriptions in order to retrieve candidate categories. At this stage, we do not disambiguate the retrieved categories. Irrelevant categories (with incorrect meaning with regard to input document's context) get low node feature values and are eliminated during AMN inferencing.

2.6.3 Temporal Relevance

Evolving personalized categorization system over time has two dimensions: (i) evolving the classifier when new documents with new categories (which exist in KnG) are seen by our system, and (ii) evolving C^{org} when new categories are added to KnG. For the first case, assuming that the collaboratively built knowledge graph KnG is upto- date with all the categories, our spotting phase identifies the features in the document corresponding to the new categories and adds them to the candidate categories. If these categories get label 1 during the inference, they are considered to be part of C^{org}. For the second case, the challenge lies in updating the already classified documents with the new categories added to KnG. One strategy of handling this could be to look at the neighborhood of newly added categories in KnG, retrieve the already classified documents having categories present in this neighborhood and reassign categories to these documents by repeating our inference algorithm.

2.7 Experiments and Evaluation

We now evaluate our proposed approach using various datasets and compare it against other methods. We start with describing our experimental setup, evaluation methodology,

and finally discuss our results.

2.7.1 Global Category Catalog (GCC)

We extract Wikipedia Category/Page titles and add them to our GCC. We also construct a description text for each category in the GCC from the first few paragraphs (gloss) of Wikipedia's page. In the case of Wikipedia categories, we concatenate glosses from multiple pages that are classified under that category. We also carry over the structural constructs that are available in Wikipedia (such as inlinks, outlinks, titles, and the like) to our GCC.

2.7.2 Datasets

Although several benchmark text classification datasets (such as 20NewsGroups, Ohsumed, Reuters, and the like) exist, our choice of datasets was based on the existence of at-least 100 class labels in the dataset. We report experiments on one existing benchmark dataset and another manually curated dataset, described below.

2.7.2.1 Reuters RCV1-v2

The Reuters RCV1-v2 collection consists of 642 categories and a collection of 804,414 documents that are multi-labeled. These documents are further classified into a training set that consist of 23,149 documents and a test set with 781,265 documents. We find the category set of the Reuters RCV1-v2 collection to be quite coarse-grained.

2.7.2.2 Technical Documents From arXiv.Org

The arXiv is an archive for electronic preprints of scientific papers in various fields which can be accessed online. Using the Amazon S3 service, we downloaded approximately 600 documents from arXiv in the field of computer science. The purpose of this experiment was to evaluate the fine-grained class labels that are obtained from the GCC. A group of human labelers manually labeled these documents and we evaluated the performance of our system.

2.7.3 Evaluation Methodology

We report our results in two different settings: (i) a warm start setting and (ii) a cold start setting.

2.7.3.1 Warm Start Experiments

In this setting, we assume that the librarian has a fair idea of the categories that she needs and has identified them apriori. Such a setting helps us demonstrate how, on a standard classification dataset, the Markov network helps to propagate inferred information from a category to other *related* categories. We performed warm start experiments on the Reuters RCV1-v2 collection. While the AMN model exploits inter-class correlation, the per-class SVM model incorporates feedback on document labels more directly. In the absence of an inter-class correlation, our model degenerates to a multi-class SVM classifier. To demonstrate this, we report experiments with two different strategies for sampling documents: (i) clustered sampling and (ii) random sampling.

Experiments on correlated categories (Clustered Sampling) In this setting, we selected 66 pairs of *related* Reuters categories,which spanned 96 categories. For *example*, the categories MANAGEMENT and MANAGEMENT MOVES are *related*. So are LABOR and LABOR ISSUES. Two categories were considered to be *related* if the number of training documents that carry both labels, exceeded a certain threshold. Our clustered sampling entailed sampling of documents that were labeled with both categories in any of the 66 pairs. We picked 5000 training documents and 2000 test documents using this clustered sampling procedure. We further divided the training set into 100 batches of 50 documents each. We iterated through the batches and in the k^{th} iteration, we trained our model (SVMs, γ_i, AMN feature weights) using training documents from all batches upto the k^{th} batch. For each iteration, we performed AMN inference on the sample 2000 test documents. In Figure 2.6, we report the average F1 score of the test sample for each of the 100 iterations.

The curves that are labeled EVO-* correspond to the F1 score of our system whereas the curves labeled mSVM-* are for the F1 score of a plain multi -class SVM. These graphs

are plotted for experiments with different choices (0.1, 1, 10, 100) of the SVM hyper-parameter C. As we expect, after approximately 20 iterations, some of the better-trained SVMs start propagating their learning to their correlated neighbors (enforced by AMN), thus boosting the overall F1 score. In other words, the learning rate in our model is faster than that of a plain SVM model. Hence, with fewer training examples we can achieve the same level of accuracy/recall as that of a plain multi -class SVM that is trained on significantly more examples.

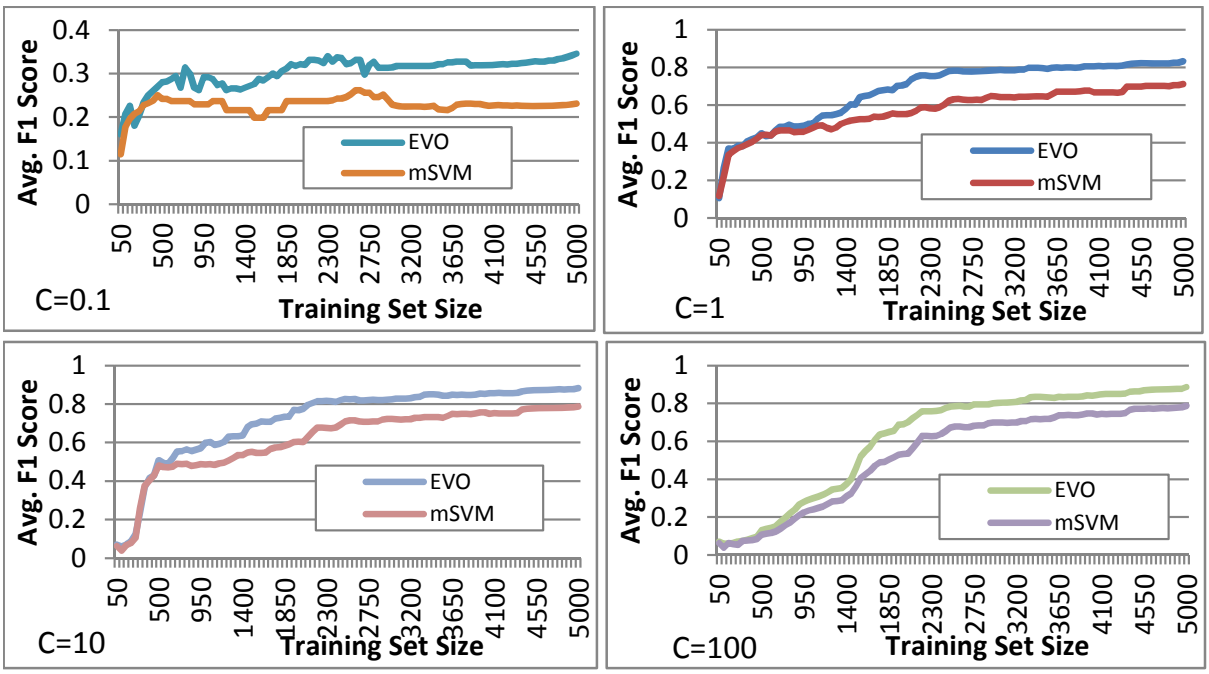
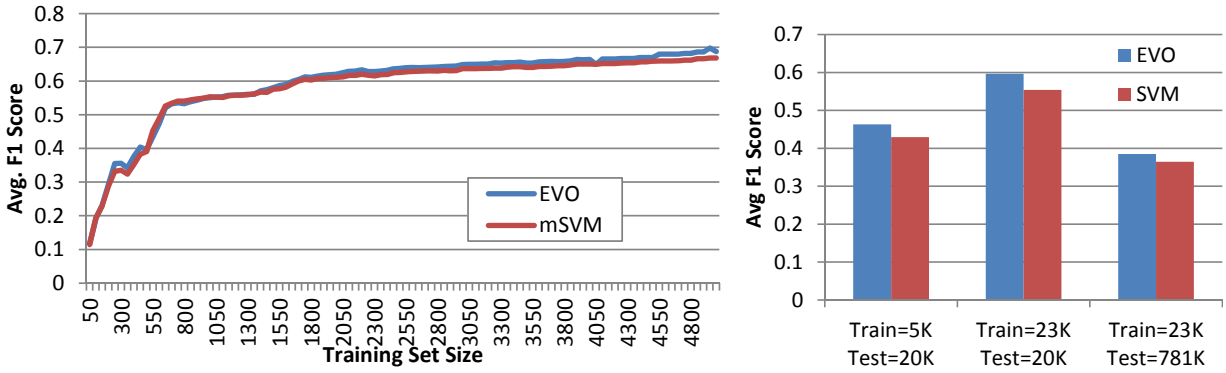


Figure 2.6: Comparison of average (macro) F1 scores of our system (EVO) with SVM on different c values

Experiments on uncorrelated/loosely correlated categories (Random Sampling) When there is no correlation or very little correlation between the categories, the AMN will not contribute much to the inference. To study this case, we selected all the Reuters categories and randomly sampled 2000 test documents from the Reuters standard test split and 5000 training documents from the training split.The rest of the evaluation procedure is the same as that in the previous case.

Figure 2.7a shows the average F1 score over 100 iteration. Since there is not much correlation between the categories, in most of the iterations, F1 score of our model (EVO) follows the F1 score of multi -class SVM (mSVM). Due to the presence of small number



(a) Increasing Training Set Size

(b) Aggregate

Figure 2.7: Comparison of avg (macro) F1 scores of our system (EVO) with SVM (random sampling)

of correlated categories in the test set, we see a small increase in the F1 score after approximately 40 iterations.

Figure 2.7b depicts the average F1 scores of EVO and mSVM over the entire Reuters collection of test and train documents. EVO performs approximately 2 to 5% better.

2.7.3.2 Cold Start Experiments

In this setup, we assume that the librarian does not have any predefined categories to start with and would like to adapt the categories from GCC. We evaluated this case by sampling 263 technical documents from different streams of Computer Science in the arXiv publications repository. With the help of eight human annotators, we assigned categories to each document by using the vocabulary of Wikipedia article names. Each annotator annotated approximately 32 documents. These annotators were asked to assign between 1 to 15 categories for each document with the category vocabulary (Wikipedia article names) limited computer science domain. On an average, they assigned 6.5 categories per document.

We carried out five fold cross validation with 210 training documents in each fold and 53 test documents. In each fold we trained our model (SVMs, γ_i, AMN feature weights) using the training set and evaluated consistency, precision and recall on the test set. During the training phase, we also applied localization techniques by which we recorded feedback for the system suggested categories in three forms:“Correct”,“Incorrect” and “ Never again”.

Number of Documents	263
Total Categories Discovered by Human Labelers	1054
Total Categories Discovered by EVO	819
Common Categories between EVO and Human Labelers	353
Total Categories Discovered by Wikipedia Miner	1943
Common Categories between Wikipedia Miner and Human Labelers	368
average Consistency over all docs by EVO	**37.69%**
average Consistency over all docs by Wikipedia Miner	24.56%

Table 2.4: Cold Start experiment results and comparison with WikipediaMiner

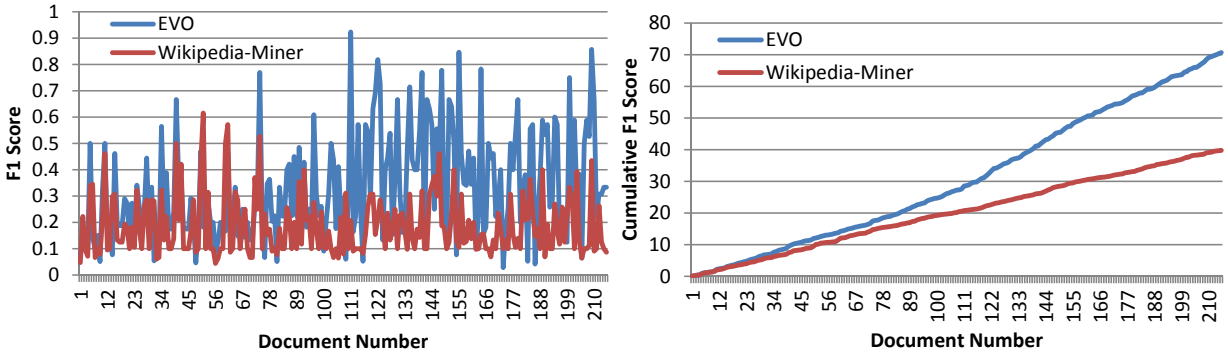
We measured the consistency [116] as:

$$Consistency = \frac{2|A \cap B|}{|A| + |B|}$$

where A and B are the total number of categories that the two systems assign (in this case, one is our system and the other is the human labeler.)

Table 2.4 shows the overall consistency of our system with human annotators. We have also compared the consistency with the Wikipedia Miner system. Note that, in some cases WikipediaMiner may generate a category that is relevant to the input document, but out of the computer science domain. We have treated such labels as incorrect because, we are interested in evolving an organization specific categorization system, which is the goal of this chapter.

In Figure 2.8 we compare the F1 scores of all the documents with WikipediaMiner. In this experiment, we picked up arxiv documents one by one, generated categories and recorded user feedback for 10 categories. We computed the F1 score for each document by considering the number of categories that were retrieved by our system and those that were entered by the human annotators. As is evident from Figure 2.8, our system performs better than Wikipedia Miner due to its learning ability and propagation of learning to



(a) Documentwise F1 score

(b) Cumulative F1 Score

Figure 2.8: F1 score comparison of EVO with WikipediaMiner

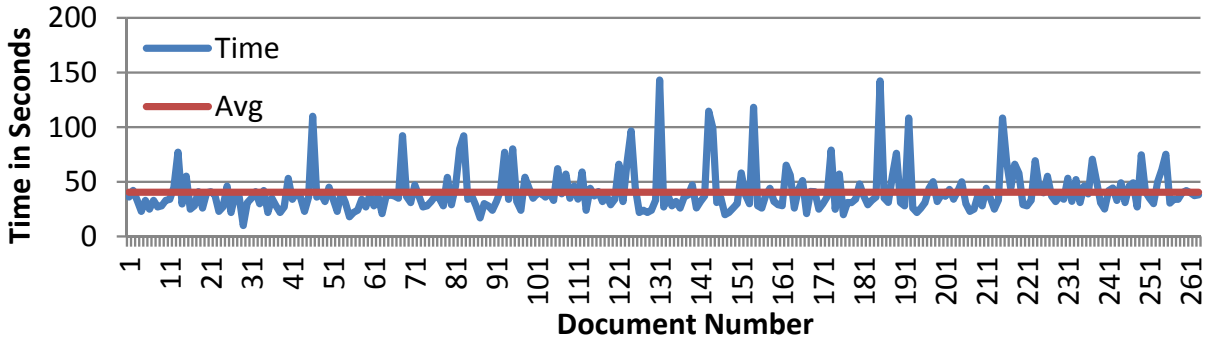


Figure 2.9: Time measurements for EVO

neighbors over AMN. (The cumulative F1 score in 2.8b is the sum the of F1 scores of all documents)

Figure 2.9 shows the time required by EVO to generate the categories of each of the documents.

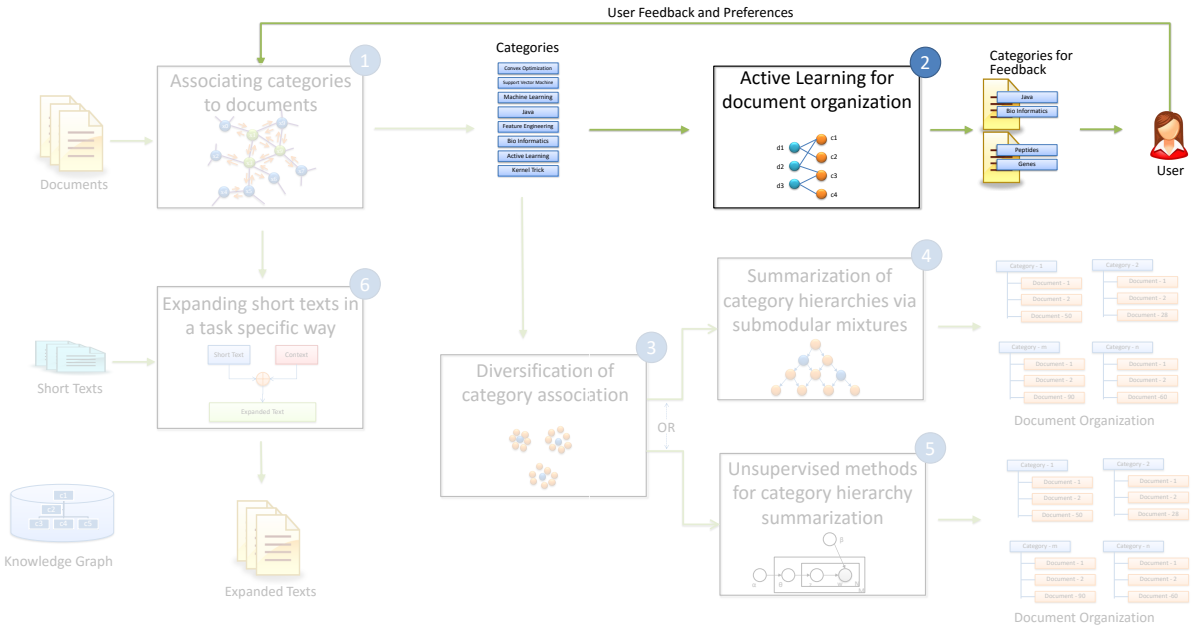
We also tried to compare our work with CTM [16], but CTM took an unacceptably long time (approximately 90 hours) to converge in a setting in order to discover approximately 1000 categories. Most of the categories that were discovered by CTM did not make much sense. Hence we do not consider CTM and similar unsupervised techniques to be suitable to solve our problem.

2.8 Conclusion

In this chapter, We presented an approach for the evolving of an organization specific multi-label document categorization system by adapting a global category catalog in to a local document collection. The resulting catalog not only fits the documents in the digital library, but also caters to the perceptions of users in the organization. We address this by learning an organization specific document categorization meta-model using an Associative Markov Networks by blending the static features that exploit the structural similarities between the categories in the global category catalog and the input document and the dynamic features that help in learning discriminative models such as SVM in an AMN setup along with user defined constraints that help in localization of the global category catalog. In the process we also curate the training data.

Chapter 3

Active Learning for Categorization*



*Published: Bairi, R. B., Vani, A., Ahuja, P., & Ramakrishnan, G. (2015, March). Categorizing videos using a personalized category catalogue. In Proceedings of the Second ACM IKDD Conference on Data Sciences (pp. 49-58). ACM.

Chapter Summary

Evolving a document categorization (described in previous chapter) involves the training of the system with user feedback ("correct", "incorrect", and "never again") as explained in the previous chapter. This training can pose a large cognitive load on the user, if the user has to provide feedback on every category of every document. To reduce this cognitive load and to achieve greater accuracy with fewer training labels (that is, feedback) we present a novel joint Active Learning technique in this chapter,in order to jointly select document and category pairs for feedback. Unlike previous approaches in which active learning is used to select a set of documents to seek labels, our approach selects a set of document-category pairs to seek feedback. That is, it jointly selects documents and categories that are most useful in seeking feedback. We adopt uncertainty based techniques for active learning which have been successfully used in many applications in the literature. We propose many techniques to identify the most uncertain categories for a document from the Markov Network which is constructed as explained in the previous chapter. We compare our joint active learning model with other uncertainty based models that are proposed in the literature and show that it performs better in learning to categorize documents with fewer feedbacks/labels thus reducing the cognitive load on the users.

3.1 Introduction

In the previous chapter we presented a technique using the AMN framework to automatically organize digital documents with the help of a global category catalog. The algorithm assigns a set of categories to each document by running MAP inference on the category AMN. During the process, some noisy categories are assigned to the document due to the following reasons: (i) When the categorization system is initially deployed, we believe that the dynamic node features of AMN (which consists of scores from many different classifiers) would not have been sufficiently trained. Hence, the dynamic feature values might not initially provide reliable signals in deciding the relevance of a category to the input document. This leads to the noisy category assignment. (ii) Until the dynamic node features are trained for the users interests/preferences, they can score the categories as relevant to the documents, even though the user does not wish to have those categories. We consider such categories also as noisy categories, even though they are relevant to the contents of the document. In order to minimize the assignment of noisy categories to the documents, we need to train the dynamic features (essentially, the classifiers at the nodes that provide classification scores as dynamic feature values) by providing feedback (which acts as labels) to indicate if the assigned categories are "relevant" or "not relevant" to the documents.

Providing feedback involves inspecting every category that is assigned to every document and marking it "relevant" or "irrelevant". This is a tedious and resource -intensive (time and human effort) job. In order to reduce the cognitive load, researchers have developed "active learning", which is a family of machine learning methods that may prioritize (or query) the data instances to be labeled for training by a human annotator (or an oracle). Research has shown that *active learning* can achieve higher accuracy with fewer labeled examples than *passive learning* (commonly known as *supervised learning.*)

In passive/supervised learning, the goal of a learner is to infer an accurate predictor from labeled training data. The labeled training data are examples of input-output pairs (x, y): the output (or label) y represents the correct answer to a question that is associated with the input x. For example, in our categorization problem, the label y is the "relevant" or "not relevant" answer to whether a particular category x which is assigned to the

document is correct or not. These labeled examples are collected prior to the learning (training) process. The intention is to deploy a learned predictor to predict the labels of input instances x that are encountered in the future. The goal of the learner, during the training process, is to infer such a predictor from the training data which is accurate with respect to these future input instances.

Active learning models a slightly different framework in which the initially available data does not have any labels, that is, each training data point is simply an input x without an associated label y. The goal of the active learner is the same as that of a passive learner: to infer an accurate predictor of labels from inputs. However, the active learner is allowed to request the label y of any particular input x in the training data. These requests can be made sequentially, so as to adapt to the results of previous label requests. In our categorization problem, this function of the active learner can be considered to be a request to the to identify user whether a particular category assignment to a particular document is correct or not[1]. This interactive process of building up a (partially) labeled data set may continue for some time, but eventually a predictor (dynamic features, that are the node specific classifiers in the AMN model) must be returned by the active learner for use in predicting the labels of future input instances.

The most commonly used query framework in active learning is uncertainty sampling [83]. In this framework, an active learner queries the instances for which it is least certain in the aspect of the manner of labeling. This approach is often straightforward for probabilistic learning models [125]. Several of the proposed algorithms [141, 58, 97] adopt uncertainty based principles for active learning. Dan Roth *et al.* [117], present global and local margin based techniques for active learning in the structured output spaces with multiple interdependent output variables. Aron Culotte *et al.* [38], present a new active learning paradigm which not only reduces the number of instances that the annotator must label, but also assesses how difficult each instance is to annotate.

Due to the simplicity and effectiveness of the uncertainty sampling technique, we adopt this method for active learning in our problem. We propose several strategies for the active learner to identify the most uncertain categories for a document using the AMN

[1]It is also possible to ask the user to label the whole document. For each such labeled document, we can store category-document assignment as correct.

framework that is described in the previous chapter. We then develop a new technique for joint active learning on the document space and the category space, in which the system identifies the most uncertain document-category pairs as a query and seeks user feedback the for it. Based on this feedback, the system learns a model for categorize the documents under the right categories (honoring user preferences/interests) from the global category catalog. In the following sections, we formally describe our active learning approach.

3.2 Deciding Uncertain Categories

We will first define some basic concepts and then delve deeper into our techniques for active learning.

- Uncertain Node: A category node C_i in AMN is more uncertain than node C_j if $|pr\,(y_i = 1) - pr\,(y_i = 0)| < |pr\,(y_j = 1) - pr\,(y_j = 0)|$.
- Influencing Node: A category node C_i in AMN is an influencing node if there exists a node C_j in the neighborhood of C_i such that $y_j = k$ if $y_i = k$ for $k \in \{0, 1\}$.
- Inveigled Node: A category node C_i in AMN is an inveigled node if there exists a node C_j in the neighborhood of C_i such that $y_i = k$ if $y_j = k$ for $k \in \{0, 1\}$.

3.2.1 Notion of Label Flipping

Let $Y = [y_1 \cdots, y_i, \cdots y_n]$ be the labeling of all n nodes in the AMN that yields the solution to the inference problem in (2.6). Each $y_i = 0$ or 1. Consider flipping the label of the i^{th} node to $\overline{y}_i$, where $\overline{y}_i = \begin{cases} 1 & \text{if } y_i = 0 \\ 0 & \text{if } y_i = 1 \end{cases}$. Let $Y_i = [y'_1, \cdots y'_i = \overline{y}_i, \cdots y'_n]$ be the labeling that is obtained by re-running the inference in equation (2.6) after this flip.

The quantity $\triangle Y_i = n - \sum_{i=1}^{n} \delta\,(y_i, y'_i)$, (where δ is the Kronecker delta function) indicates the number of label flips in Y_i with respect to Y which is a result of the flipping of the label of the i^{th} node.

If the flipping of label y_i in Y results in flipping of y_j in Y_i, then there exists a path in the MN from node C_i to C_j on which every node's label is flipped in Y_i. That is, if C_k is a node on such a path, then $y'_k = \overline{y}_k \ \forall k$.

Let C_j be the node the label of which was flipped due to the flipping of C_i's label. We know that due to the Local Conditional Independence property of MN, $C_j \perp C_{V \setminus \{j\} \cup nbr(j)} \mid C_{nbr(j)}$ where $nbr\,(j)$ is the set of neighbors of node j; the label of node C_j will change only if one of its neighbors' labels changes. Let C_k be one of neighbors. We can recursively apply the same argument for C_k. Eventually we should get the node C_i in this recursion, which indicates that there is a path.

Computing Y_i for every node is computationally expensive. We need to run MAP inference once for every node after fixing its label. Hence we present an approximated algorithm. For each node we flip its label and estimate if this results in flipping of any of its neighbors' labels. If it does, then we repeat the procedure for all the neighbors' whose labels have flipped. Algorithm 3 outlines this procedure. Empirically we have found that the approximation error is less than 5%.

A feature space and a hyperplane in separates instances (nodes) with label 1 from instances that have label 0 and pass through the origin.

Consider a node C_i that is labeled 1 after the MAP inference. Let $Nbr_0\,(C_i)$ be the neighbors of C_i which are labeled 0 and $Nbr_1\,(C_i)$ be the neighbors that are labeled 1. We know that $\mathbf{w}^1.\mathbf{x}_i + \mathbf{w}^{11}.\sum_{j \in Nbr_1(C_i)} \mathbf{x}_{ij} \geq \mathbf{w}^0\mathbf{x}_i + \mathbf{w}^{00}.\sum_{j \in Nbr_1(C_i)} \mathbf{x}_{ij}$. This is due to MAP inference at node C_i. Otherwise C_i would have been assigned label 0. Using simple algebraic manipulations, we can re-write this expression as $\overline{\mathbf{w}}.\overline{\mathbf{x}_i} \geq 0$, where $\overline{\mathbf{w}} = \overline{\mathbf{w}^1} - \overline{\mathbf{w}^0}$, $\overline{\mathbf{w}^1} = [\mathbf{w}^1; \mathbf{0}; \mathbf{w}^{11}]$, $\overline{\mathbf{w}^0} = [\mathbf{w}^0; \mathbf{w}^{00}; \mathbf{0}]$ and $\overline{\mathbf{x}_i} = \left[\mathbf{x}_i; \sum_{j \in Nbr_0(C_i)} \mathbf{x}_{ij}; \sum_{j \in Nbr_1(C_i)} \mathbf{x}_{ij}\right]$. Note that we use the symbol ';' to concatenate the vector elements according to Matlab notation. $\mathbf{0}$ is a vector of zeros. The expression $\overline{w}.\overline{\mathbf{x}_i} \geq 0$ represents the half space that is separated by the hyperplane which is specified by $\overline{w}$, and passes through origin in the feature space of $\overline{\mathbf{x}_i}$. Similarly we can show that the node $\mathbf{x}_i$ labeled 0, resides in the half space $\overline{w}.\overline{\mathbf{x}_i} \leq 0$.

If a node C_i has higher $\triangle Y_i$ than a node C_j, then C_i is more influencing than C_j.

Algorithm 3 Flip-Test Score Generation

function FLIP_TEST_SCORER($C, \mathbf{y}, \mathbf{w}$)

Input : AMN of categories $C = (C_1...C_n)$, inferred labels $\mathbf{y} = (y_i, ..., y_n)$, feature weights $\mathbf{w} = (w_n^0, w_n^1, w_e^0, w_e^1)$

Output : Flip-Test score for every node

Initialize $R[C_i] = 0 \ \forall i = 1..n$

for all $C_i \in C$ **do**

Initialize queue Q with C_i

Initialize $V = \{\emptyset\}$

$R[C_i] = 1$

while Q **not** *Empty* **do**

$T = Q.pop()$ ▷ Pick next item from queue

Flip the label of T

for all $N_i \in Nbr(T)$ **and** $N_i \notin V$ **do**

Let z_i = label of N_i and $\overline{z}_i = \sim z_i$, flipped label of N_i

$$\phi = w_n^0.x_i.\delta(z_i, 0) + w_n^1.x_i.\delta(z_i, 1) + \sum_{j \in Nbr(N_i)} \Big(w_e^0.x_{ij}.\delta(z_i, 0).\delta(z_j, 0) + w_e^1.x_{ij}.\delta(z_i, 1).\delta(z_j, 1) \Big)$$

$$\overline{\phi} = w_n^0.x_i.\delta(\overline{z}_i, 0) + w_n^1.x_i.\delta(\overline{z}_i, 1) + \sum_{j \in Nbr(N_i)} \Big(w_e^0.x_{ij}.\delta(\overline{z}_i, 0).\delta(z_j, 0) + w_e^1.x_{ij}.\delta(\overline{z}_i, 1).\delta(z_j, 1) \Big)$$

if $\overline{\phi} > \phi$ **then**

Flip the label of N_i

$R[C_i] = R[C_i] + 1$

$Q.push(N_i)$ ▷ Add N_i to queue

end if

end for

Add T to V

end while

end for

return R

end function

If we imagine a hyperplane as in Theorem 3.2.1, all the categories that have label 0 are on one side of the hyperplane and those that have label 1 are on the other side. Nodes that are closer to the hyperplane are considered to be more confusing than the one that are far away. Nodes that are close to the hyperplane that has edges with other nodes, that are also close to but on the same side, are more likely to pull over the neighbors to the other side of the hyperplane if those nodes are moved to the other side of the hyperplane (that is, flipping the label). On the other hand, nodes that are far away from the hyperplane are less likely to pull over the neighbors along with them because those neighbors themselves have strong node potentials.

If a node C_i labeled 1 (or 0) in AMN has strongly associated neighbors that are labeled 0 (or 1), it is more likely to be an uncertain node than a node with neighbors that have identical labels.

Consider two nodes C_i, C_j with a strong edge between them. Case 1: Let both of them have the same label, for example 1. Let $p = |pr\,(y_i = 1) - pr\,(y_i = 0)|$. We can estimate p in terms of potentials as $p \propto |(\mathbf{w}_n^1 - \mathbf{w}_n^0)\,.\mathbf{x}_i + \mathbf{w}_e^1.\mathbf{x}_{ij}|$. Case 2: Let both the nodes have different labels, for example $y_i = 1, y_j = 0$. In a manner similar to the previous case, we can estimate $p' \propto |(\mathbf{w}_n^1 - \mathbf{w}_n^0)\,.\mathbf{x}_i|$. Clearly, $p' < p$. According to Definition 3.2, nodes in Case 2 are more uncertain than in Case 1.

A node with $y_i = 1$ and $log\varphi\,(x_i, 1) < log\varphi\,(x_i, 0)$ is more likely to be an inveigled node. So is a node with $y_i = 0$ and $log\varphi\,(x_i, 0) < log\varphi\,(x_i, 1)$.

Consider an AMN without any edges. If $log\varphi\,(x_i, 1) < log\varphi\,(x_i, 0)$,then MAP inference assigns label 0 to it. However, when edges are considered, MAP inference assign label 1, that is, $y_i = 1$. This implies that the node C_i's label changes on the basis of its neighbors' labels. As per Definition 3.2, C_i is an inveigled node.

A node that has a lower margin distance

$$m_i = \left| \left(w^1.\mathbf{x}_i + w^{11}.\sum_{j \in Nbr_1(x_i)} \mathbf{x}_{ij} \right) - \left(w^0\mathbf{x}_i + w^{00}.\sum_{j \in Nbr_0(x_i)} \mathbf{x}_{ij} \right) \right|$$

is more uncertain.

Consider the hyperplane $\overline{w}$ as defined in Theorem 3.2.1. The distance of a point $\overline{x_i}$ from this hyperplane is given by $\frac{\overline{w}.\overline{x_i}}{\|\overline{w}\|^2}$. By ignoring the denominator (because it is common for

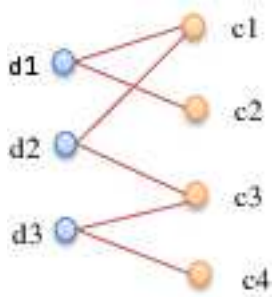


Figure 3.1: Document Category Bipartite Graph

all points) we have the distance $\propto \overline{w}.\overline{x_i}$. By expanding and rearranging the terms, this expression reduces to m_i. Therefore,the lower the m_i,the closer is the point to hyperplane $\overline{w}$, and hence more uncertain.

Our active learning strategy is to seek feedback for nodes that are influencing, inveigled, and uncertain.

We prepare a ranked list of nodes (with the most uncertain node at the top) to seek feedback. Influencing and Inveigled nodes are placed at the top of the list. If there are many such nodes, we order them according to the margin distance m_i.

3.3 Deciding Uncertain Documents

In the previous section, we identified the most uncertain categories for each document to be categorized. We now make use of this to identify the most uncertain documents.

The association between documents and categories can be represented as a bipartite graph as is shown in Figure 3.1. Documents are shown on the left side and categories on the right side. Each document is connected to its L most uncertain categories.

Our approach is to select a subset of documents that results in the maximum coverage of the most uncertain categories. Specifically, we solve the following optimization problem to identify the uncertain documents.

$$\underset{\mathbf{y},\mathbf{z}}{\text{argmax}} \quad \sum a_i y_i + \sum b_j z_j \tag{3.1}$$

$$\text{s.t.} \quad \sum z_j = P \tag{3.2}$$

$$\sum z_j \geq y_i \quad \forall i \text{ connected to } j \tag{3.3}$$

$$0 \leq z_j \leq 1 \tag{3.4}$$

$$0 \leq y_i \leq 1 \tag{3.5}$$

$$\forall i \in I \text{ and } j \in J$$

where, I is the set of indices of categories J is the set of indices of documents, and:

- a_i is the gain that is associated with selecting the i^{th} category C_i. We choose this to be the maximum uncertainty score of C_i. The uncertainty score of C_i is the *margin distance* [141] from the margin that is introduced in Theorem 3.2.1.
- b_j is the gain that is associated with selecting the j^{th} document d_j. We choose this to be the uncertainty score of $d_j = f(C_1, ..., C_k)$; for some function f of related categories. For *for example*, a simple version of f can be the one that chooses the score of the most uncertain category that is connected to d_j.
- $z_j \in \{0, 1\}$ and $y_i \in \{0, 1\}$ will be the integer solution at optimality.
- P is the number of documents for which the user is willing to give feedback.

Feedback is sought from the user for the documents with $z_j = 1$. Note that for each document, feedback is sought only for those categories that are identified as the most uncertain in Section 3.2.

3.4 Incorporating Feedback for Personalization

Based on the feedback provided by the user, we learn and update the per-class SVM models. For instance,the user may mark category C_i as correct for document d and C_j

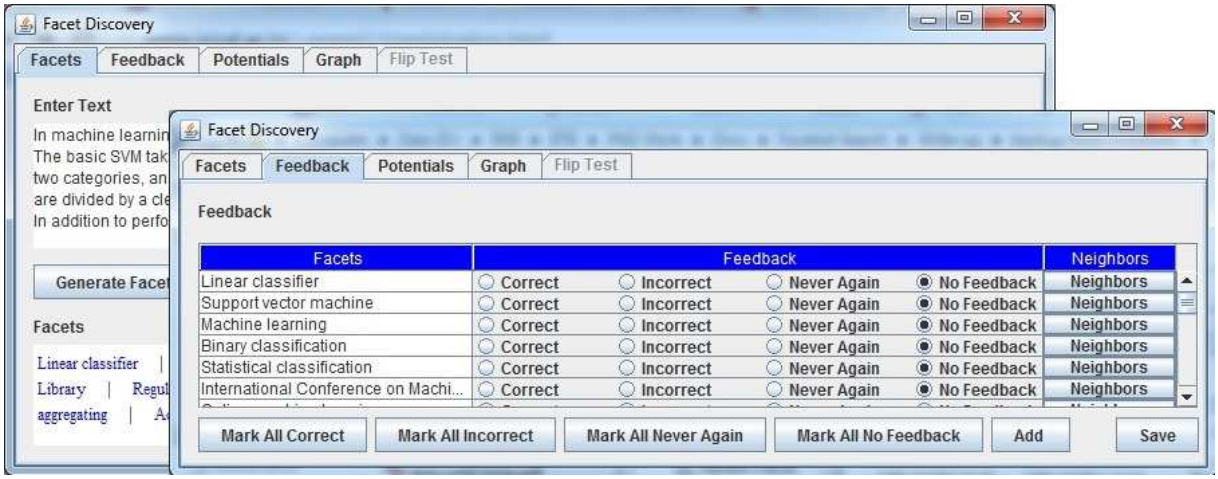


Figure 3.2: Interface for category generation and feedback

as incorrect. We subsequently treat document d as a positive example for C_i and as a negative example for C_j. We update Svm_{C_i} and Svm_{C_j} by including d as an additional training example. This gives us updated SVM parameters $(\mathbf{w}_{C_i}, b_{C_i})$ and $\left(\mathbf{w}_{C_j}, b_{C_j}\right)$ which we incorporate in the SVM decision functions (the dynamic node features described in Section 2.4.1.1) for subsequent categorization. Periodically, we also retrain the AMN so that the weights for the SVM features are recomputed as the individual SVM models mature and start to stabilize.

Figure 3.2 presents snapshots of the interface of our system in order to solicit user feedback. In the snapshot on the left side, the user can enter text (or choose one from a collection) and ask the system to suggest candidate categories. In the snapshot on the right side, the system lists categories for feedback in decreasing order of uncertainty. User can provide feedback by choosing one of the options: "Correct", "Incorrect", "Never Again". (Note, in the GUI, we refer to categories as *Facets*)

3.5 Experiments and Evaluation

In this section we evaluate the effectiveness of our proposed active learning method by comparing it against other methods from the literature and show that our method is superior in the aspect of the evaluation metric.

3.5.1 Preparation

To evaluate our proposed method of active learning, we construct an AMN over categories for each document using the techniques that are presented in the previous chapter. As in the last chapter, we report our experimental results of the Reuters RCV1-v2 collection. This collection consists of 642 categories and a collection of 804,414 documents that are multi-labeled. These documents are further classified into a training set that consists of 23,149 documents and a test set with 781,265 documents.

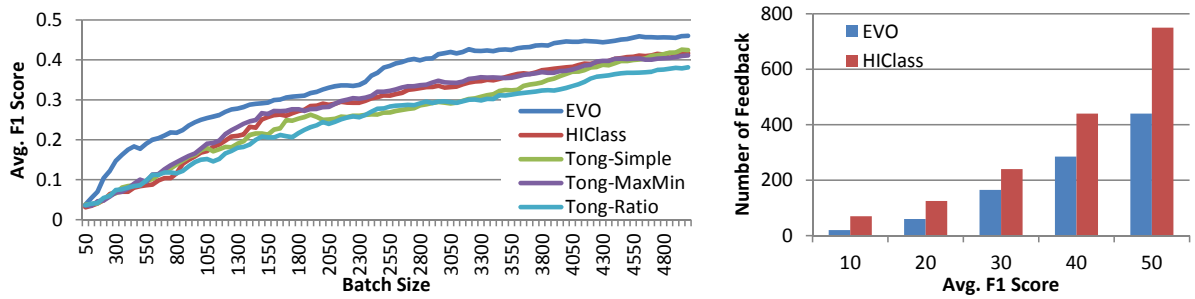
We construct the AMN over categories for each Reuters document using the GCC that is built from Wikipedia as explained in the previous chapter. These AMNs are used to validate our active learning techniques that are explained in Sections 3.2 and 3.3.

3.5.2 Evaluation Methodology

We compare the effectiveness of our active learning algorithm with the HIClass system [58] by Godbole et al. and Simple-Margin, MaxMin-Margin, Ratio-Margin algorithms by Tong et al. [141]. The techniques proposed by Tong et al. are based on the margin distance in a two class SVM classifier. We extend this to a multiclass scenario by aggregating the margin distances for multiple classes.

In these evaluations, as in our warm start tests, we obtained 5000 training documents randomly from Reuters training split and divided them into 100 batches of 50 documents each. In each iteration, we added the next batch and trained our model. However, the feedback was recorded only for 10 most uncertain documents and five most uncertain categories for each document as suggested by our active learning algorithm. Based on the feedback, the documents were treated as positive or negative examples for the categories and the underlying SVMs were retrained. Thereafter, we retrained the AMN model parameters.

In cases of HIClass, Simple-Margin, MaxMin-Margin and Ratio-Margin, in each iteration, we obtained 10 most uncertain documents from the current batch as suggested by the respective algorithms. For all those documents we picked up five categories with the lowest



(a) F1 score comparison

(b) Feedback comparison

Figure 3.3: Comparing our Active Learning technique with others

SVM decision values as the most uncertain categories. Further we recorded feedback and retrained our model parameters as in the previous case.

3.5.3 Result Discussion

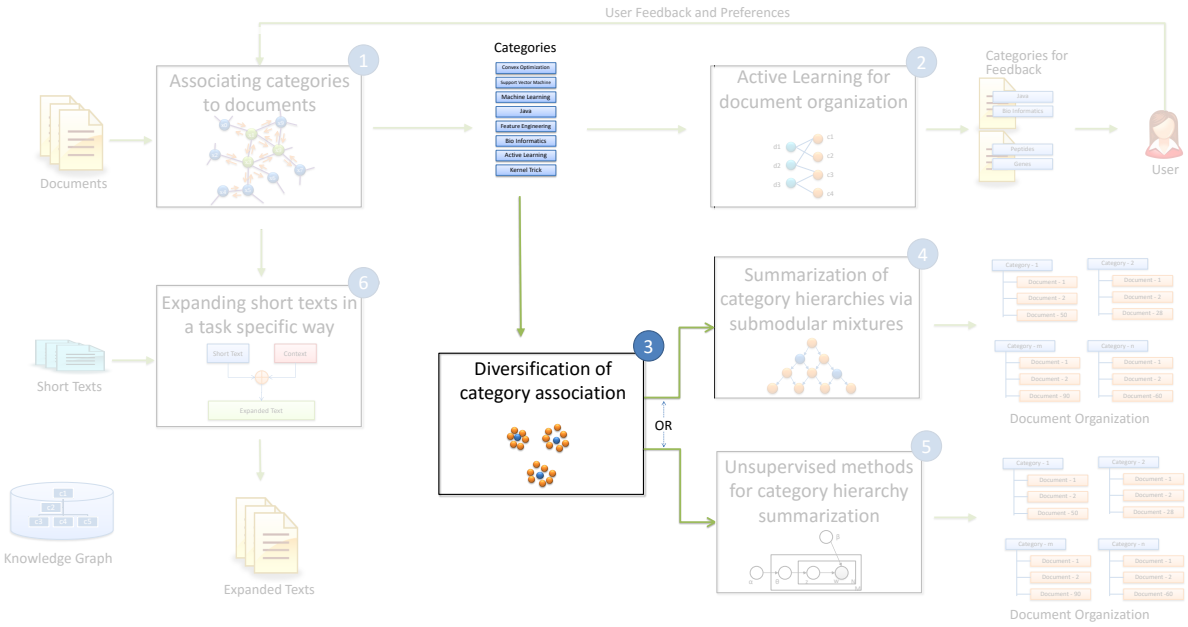
From the results in Figure 3.3a we observe that, with our combined active learning logic over the document and category space, we are able learn at a faster rate and with an improved F1 score. Viewing the results from the perspective of the amount of feedback needed to achieve a required level of F1 score, we observe in Figure 3.3b that, with our technique, we need a significantly lower amount feedback, resulting in a lowers cognitive load on the user.

3.6 Conclusion

We presented an approach for joint active learning in the document-category space by using uncertainty sampling techniques. We showed how uncertain, influencing and inveigled nodes can be identified in a category AMN, from which the most uncertain categories can be presented as a query to the user. Instead of querying the users for the uncertain categories for each document, we presented a joint active learning technique to identify the most uncertain documents and categories together as document-category pairs. Our experimental results of the Reuters RCV1-v2 document collections showed that our proposed active learning approach outperforms several other baselines and methods.

Chapter 4

Diversification of Category Assignment*



*__Published__: Learning to Generate Diversified Query Interpretations using Biconvex Optimization, Ramakrishna Bairi, Ambha A, Ganesh Ramakrishnan., Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 733-739, Nagoya, Japan, October 2013, Asian Federation of Natural Language Processing.

Chapter Summary

With the Associative Markov Network (AMN) based framework for category assignments to the documents (introduced in Chapter 1 and 2), many categories that are closely related to each other get assigned to a document. This is due to the "associative" property of AMNs. Even though the categories are relevant to the document, a large set of similar categories make it harder to bring out the diversity in the category distribution of the document. Moreover, when there is a budget (K) constraint on the number of categories that can be associated to a document, it becomes extremely important to produce a diverse but relevant set of categories in the precious top K positions. This calls for addressing two types of needs: (i) producing relevant categories for documents and (ii) selecting a set of K diverse categories to satisfy different classes of information needs. In this chapter, we present a novel technique using a Biconvex optimization formulation as well as adaptations of existing techniques from other areas, for addressing these two problems simultaneously. We propose a graph based iterative method to choose diversified categories. We evaluate these approaches on the QRU (Query Representation and Understanding) dataset used in SIGIR 2011 workshop as well as on the AMBIENT (Ambiguous Entities) dataset and present results on generating diversified query results. We also compare these approaches against other online systems such as *Surf Canyon, Carrot2, Exalead and DBpedia* and empirically demonstrate that our system produces competitive results.

4.1 Introduction

The Associative Markov Network (AMN) based category identification results in association of bunch of similar categories to a document. This is due to the "associative" or "attractive" edge potentials property of the AMN. The MAP inference on AMN encourages strongly related neighbor nodes in the graph to attain similar labels. Hence, a node that is labeled 1 strongly influences its related neighbors also to get label 1. Due to this, a digital document usually is assigned with many categories of similar nature. For example, This poses a few challenges for the automatic document classification system. (i) A large group of similar categories naturally eclipse the smaller groups and suppress them from being evident on the documents. (ii) When the categories are used by downstream applications/stages (for example, categories are used for summarization in the next Chapter) similar categories provide restricted information for the subsequent consumers (iii) When there is a budget constraint on the number of categories assigned to a document, and similar categories provide redundant information content;then the assigned categories should be diverse enough to cover the topics of the document. This calls for presenting a diversified but relevant set of categories in the top k positions. Note that, in this chapter we consider each category assigned to a document as describing some aspect (concept) related to the document.

We present an original method as well as adaptations of some existing methods to solve this problem. In our proposed method, we construct a category graph with the categories as its nodes and edges indicating their similarity. This graph is same as the graph created in the earlier chapter (Chapter 2). We define two sets of similarity functions; one set of similarity functions measure the similarity between category nodes and the document, and another set of similarity functions measure the similarity between pairs of category nodes. During the training step, we learn a function of the feature weights to combine the features. One possibility is to apply AMN learning technique presented in Chapter 2. However, AMN training does not explicitly account for diversity in the inferred categories. Inspired by the works on GCD [48] and MMR [28], we develop a new technique for diversifying the ranking of categories. As part of this technique, we propose an algorithm – Rel-Div – to learn the node and edge weights of the category graph interactively by solving a biconvex optimization problem. At inference time, we solve a submodular optimization problem to

choose k diverse nodes and present them as the categories for the document.

Even though this chapter describes the Rel-Dev technique, one of the goals of this chapter is also to empirically study the weight learning techniques from AMN and Rel-Dev. That is, we can either use AMN training technique or Rel-Dev bi-convex optimization technique to learn the node and edge feature weights. Inferring categories of the document can then be done in two different ways: (i) do the AMN inference without budget constraint and then apply submodular optimization presented in Rel-Dev technique to choose k diverse categories; or (ii) directly apply the submodular optimization technique in Rel-Dev to the entire set of categories. Empirically we find that first approach works better due to the AMN's associative property doing a good job in first identifying all the relevant categories of the document, thereafter diversifying using Rel-Dev technique eliminates the redundant categories. In addition, the bi-convex optimization technique weight learning in Rel-Dev suffers from a local minimum problem, which deteriorates the quality of inferred categories. However, AMN weight learning objective function is convex for binary variable case and hence finds optimal weights.

In the Rel-Dev technique, we identify categories relevant to the document, yet diverse among themselves, using a set of node features – as in Chapter 2 – derived from publicly available internet encyclopedia. Though we used Wikipedia as the source, we believe that the repository can be easily extended to accommodate other catalogs like YAGO and Freebase.

We compare Rel-Dev approach with other diversification approaches (which were applied not necessarily to solve the same problem as ours) such as variants of GCD [48], Affinity Propagation [55],[56]. We evaluated results on benchmark queries from the SIGIR 2011 workshop's QRU (Query Representation and Understanding) dataset and the AMBIENT data sets. In addition, we compare the diversity results of our approach against those of other online systems such as Surf Canyon, Carrot2, Exhaled and DBpedia. Note that, in these experiments, we apply our Rel-Dev technique to the problem of diversifying search results of a query, Though this problem is different from the category diversification problem that we are dealing with, it allows us to compare our technique with baselines and other approaches on the benchmark datasets.

We summarize our contributions as: 1) Top-K diversity ranking using a graph based

approach. 2) Iterative Graph weight learning technique - A new iterative technique for learning the node and edge weights for a category graph by solving a biconvex optimization problem.

4.2 Prior Work

Most of the prior research has focused on generating diversified result urls. The approach presented by Swaminathan et al. [137] filters initial search results and covers diversified topics based on bag of words measures. Yisong and Joachims [158] train a model using Struct SVM and encode diversity as a penalty function (this is penalty for not covering certain topics). Most recently, Brandt et al. [22] and Raman et al. [114] proposed an approach for *dynamic ranking* and then group URLs with similar intentions. Debey et al. [48] formulate the problem of ensuring diversity as that of identifying relevant urls which are most likely to be visited by the random surfer. We propose a new approach for interpretation generation. A report [63] by M.A Hearst claims that clustering based on similarity measure may not always result in meaningful interpretations or labels. So, instead of dynamically generating labels, we pick labels or relevant interpretations for a query from the pool of wail able labels. We use Wikipedia as a primary source to capture these interactions along with their semantic relations. Hahn et al. [62] and Ben et al. [12] produce Wikipedia pages as search results and align the search results along a set of fine grained attributes/facets. In our work, facets (which we refer to as interpretations) are neither predefined nor necessarily fine grained. Moreover, as we will see, our interpretations need not be restricted to Wikipedia entities. Closest to our approach is the approach of Hao et al. [91]. They apply page ranking technique on the graph constructed using query log statistics to obtain diversified interactions.

4.3 Diversified Category Selection

We now formally define our problem statement and then discuss our proposed approach to solve it.

4.3.1 Diversification Problem Definition

Given a set $\mathcal{C}$ of m categories and a document d, we define a function $H(d, \mathcal{C})$ that returns a subset of categories $S \subseteq \mathcal{C}$, that are relevant to the document d. Let $S = \{c_1...c_n\}$. $H(d, \mathcal{C})$ a matching function that retrieve the categories S which are syntactically and/or semantically related to the document d. It helps reducing the search space of our algorithm. In its simplest form, $H(d, \mathcal{C})$ can just return C without performing any matching, which is not generally useful. A better form of $H(d, \mathcal{C})$ can just do a keyword based look up on $\mathcal{C}$ without applying any sophisticated retrieval technique. On the other hand, $H(d, \mathcal{C})$ can become as sophisticated as the AMN based category association technique descried in Chapter 2. Our goal is to choose a set of k categories from S and we assume that to best satisfy the topical coverage of the document, the k categories associated with the document should be diverse yet highly relevant to the document d.

4.3.2 The Training Algorithm

We expect groups of categories in S to be related to each other via some semantic relations. We initially construct a category-relation graph using $c_1...c_n$. We refer to this graph as an *Interpretation Graph*, since the categories in this graph are obtained as various interpretations of the document. While the nodes are categories from S, each edge is a relation between the categories. A relation could be one of synonymy, hyponymy, meronymy, homonymy, and the like. These relations could be obtained from external catalogs such as Wikipedia, Wordnet [100], and the like.

Each node in the graph is assigned a score which represents the relevance of the category to the document d. We use the notation b to represent the column vector (of size $n \times 1$) containing all the node relevance scores. The weight on an edge represents the degree of similarity between the two nodes connected by that edge. We use the notation C (of size $n \times n$) to represent the matrix of edge scores reflecting similarity between pairs of nodes. Note that, each column C^i of the matrix C represents an category c_i and the cell values in that column indicate the similarity of category c_i with the other categories. The scores in b are used to ensure that the subset of k categories are relevant to the document d, whereas the similarity scores in C are used to ensure diversity in the subset of k categories.

We assume that we are provided training data, consisting of documents and their correct categories. Our goals in training are to 1) develop a model for the node score b, 2) develop a model for the edge potentials C and 3) learn parameters of these models such that the set of k relevant yet diverse nodes obtained from the graph using b and C are consistent with the training data. Thus, implicit in our third goal is the following sub-problem, which is also our query time inference problem: 4) compute a subset of k best categories using b and C, that represent k diverse, but relevant categories.

4.3.2.1 Modeling Node Potentials (b)

In order to build a learning model for b, it is important to define a good set of features that characterize the node's (category's) relevance to the document. Let $N_{1..|N|}(d, S)$ be a set of $|N|$ category independent node features. Each feature $N_f(d, S)$ evaluates the relevance of categories in S to the document d and returns a vector of scores. These feature functions are problem specific and crafted carefully to bring out the relevance between document and categories (such as term overlaps, n-gram matches, and the like.).

The node potential vector b is obtained by combining the scores returned by individual feature functions $N_f(d, S)$. One of the obvious choices is to use Logistic Regression[1] [155]. That is,

$$b[i] = \frac{1}{1 + e^{-\sum_{f=1}^{|N|} w_f N_f(d,S)[i]}}$$

The weight vector $W^T = \left[w_1 ... w_{|N|}\right]$ is learned through supervised training explained in Section 4.3.2.3.

4.3.2.2 Modeling Edge Potentials (C)

To learn the edge potentials, it is important to define a good set of features that measure the similarities between every pair of nodes and return similarity scores. Higher the score, more similar are the nodes. Let $C_{1..|C|}(S)$ be the set of $|C|$ edge features that evaluate similarities between categories in S and each returns a $n \times n$ matrix of scores. These

[1]Even though other options are possible here, we restrict to Logistic Regression in this chapter.

feature functions are problem specific and crafted carefully to bring out the similarities between the categories.

The edge potential matrix C is obtained as $C = \sum_{f=1}^{|C|} \lambda_f C_f(S)$ where $0 \leq \lambda_f \leq 1$ and $\sum \lambda_f \geq 1 \; \forall f$. The weight vector $\lambda^T = \left[\lambda_1 ... \lambda_{|C|}\right]$ is learned through supervised training explained in Section 4.3.2.3.

4.3.2.3 Learning Feature Weights W^T, λ^T

We now introduce a Proposition that sets a premise for learning the feature weights. The intuition behind this approximated equality comes from the fact that, two similar categories should have similar relevance score with the document and we are interested in selecting k diverse categories.

Proposition 1:

$$b \approx \sum_{j=1}^{k} \tilde{C}^{i_j} \tag{4.1}$$

for sufficiently large k diverse categories, where, $\tilde{C}$ is the matrix $C$with the columns scaled so that the diagonal cell values match the relevance value, that is, $\tilde{C}(i,i) = b(i)$. The values $i_1 ... i_k$ represent indices of k columns of matrix $\tilde{C}$. Hence, $\tilde{C}^{i_j}$ is the i_j^{th} column of matrix $\tilde{C}$.

Let c_i be one of these k diverse categories. If the categories $c_{j_1} ... c_{j_p}$ are similar to c_i, then, $b[i] \approx b[j_1] \approx ... \approx b[j_p]$ and $C[i,i] \approx C[i,j_1] \approx ... \approx C[i,j_p] \approx 1$ and $C[t] \approx 0$, $t \notin j_1 ... j_p$. But, we know that $\tilde{C}[i,i] = b[i]$. That implies, $b[j_1] \approx \tilde{C}[i,j_1]$, $b[j_2] \approx \tilde{C}[i,j_2]$, ... $b[j_p] \approx \tilde{C}[i,j_p]$. When we take the summation on all diverse k categories, the Equation 4.1 holds.

The following example illustrates the Proposition 1. Consider the matrices b, C, and $\tilde{C}$ as shown in the Figure 4.1.

It has nine categories numbered from 1 to 9. The categories with same color code indicates that they are similar. The categories numbered $1, 4, 8$ are similar (same color coded). Similarly, categories $2, 6, 9$ and $3, 5, 7$ are similar. Suppose we want to select

C	1	2	3	4	5	6	7	8	9
1	1	0.2	0.11	0.8	0.13	0.34	0.4	0.7	0.3
2	0.2	1	0.2	0.4	0.1	0.6	0.2	0.4	0.9
3	0.11	0.2	1	0.2	0.65	0.1	0.7	0.3	0.2
4	0.8	0.4	0.2	1	0.1	0.2	0.3	0.4	0.3
5	0.13	0.1	0.65	0.1	1	0.2	0.4	0.3	0.2
6	0.34	0.6	0.1	0.2	0.2	1	0.3	0.1	0.2
7	0.4	0.2	0.7	0.3	0.4	0.3	1	0.2	0.1
8	0.7	0.4	0.3	0.4	0.3	0.1	0.2	1	0.3
9	0.3	0.9	0.2	0.3	0.2	0.2	0.1	0.3	1

b
0.6
0.2
0.5
0.7
0.6
0.1
0.4
0.9
0.3

3 2 1

$\tilde{C}$	1	2	3	4	5	6	7	8	9
1	0.6	0.04	0.06	0.56	0.08	0.03	0.16	0.63	0.09
2	0.12	0.2	0.1	0.28	0.06	0.06	0.08	0.36	0.27
3	0.07	0.04	0.5	0.14	0.39	0.01	0.28	0.27	0.06
4	0.48	0.08	0.1	0.7	0.06	0.02	0.12	0.36	0.09
5	0.08	0.02	0.33	0.07	0.6	0.02	0.16	0.27	0.06
6	0.2	0.12	0.05	0.14	0.12	0.1	0.12	0.09	0.06
7	0.24	0.04	0.35	0.21	0.24	0.03	0.4	0.18	0.03
8	0.42	0.08	0.15	0.28	0.18	0.01	0.08	0.9	0.09
9	0.18	0.18	0.1	0.21	0.12	0.02	0.04	0.27	0.3

Figure 4.1: Illustration of Proposition 1

three diversified categories, we expect one category to come from each color code. As per the Proposition 1, the first category to be selected is the category number 8. This is because, the column numbered 8 in $\tilde{C}$ has maximum similarity to the vector b. The next category to be selected is the category number 7. Because, as per the Proposition 1, the sum of columns 8 and 7 in $\tilde{C}$ has maximum similarity to the column b. Suppose we select column 4 instead of column 7, then we are not selecting diversified categories. In that case, sum of columns 8 and 4 (which have similar distribution) does not increase the similarity to the column b. Based on the similar argument, the third category selected should be category numbered 6. Thus, we select three diversified categories.

Based on the above proposition, we present an algorithm to learn weights W^T and λ^T iteratively in a supervised learning setup. The training data is provided in a vector r (of size $n \times 1$) such that $r[i] = 1$ if the category c_i is relevant to the document (and one of diverse categories), otherwise, $r[i] = 0$. Note that, the quantity $\tilde{C}r$ represents the sum of k columns (assuming k number of 1s in r) and is the RHS of Equation 4.1.

Our training objective is to learn λ^T and W^T such that Equation 4.1 holds. Formally, the problem being solved is:

$$\underset{\lambda_1 \ldots \lambda_{|C|}, w_1 \ldots w_{|N|}}{argmin} \; D\left(\frac{1}{1+e^{-\sum_g w_g N_g}}, \sum_f \lambda_f \tilde{C}_f r\right) \tag{4.2}$$

where $D(x, y)$ is a distance measure between x and y. (for example, KL Divergence, Euclidean, and the like.); $\tilde{C}_f$ is the normalized C_f as in Proposition 1.

Applying the coordinate descent technique, we learn the weights W^T and λ^T iteratively using two steps outlined in Equation 4.3 and Equation 4.4, each of them convex in the respective optimization variables, hence our optimization problem is biconvex.

div-step: Learn $\lambda_1^{(t)}, \lambda_2^{(t)}, ...$ holding $w_1^{(t-1)}, w_2^{(t-1)}, ...$ constant, by solving:

$$\underset{\lambda_1,\lambda_2,...}{argmin}\ D\left(\frac{1}{1+e^{-\sum_g w_g^{(t-1)} N_g}}, \sum_f \lambda_f^{(t)} \tilde{C}_f r\right) \tag{4.3}$$

rel-step: Learn $w_1^{(t)}, w_2^{(t)}, ...$ holding $\lambda_1^{(t-1)}, \lambda_2^{(t-1)}, ...$ constant, by solving:

$$\underset{w_1,w_2,...}{argmin}\ D\left(\frac{1}{1+e^{-\sum_g w_g^{(t)} N_g}}, \sum_f \lambda_f^{(t-1)} \tilde{C}_f r\right) \tag{4.4}$$

In *div-step*, we learn λ^T by holding W^T fixed and honoring Equation 4.1. In *rel-step*, we learn W^T by holding λ^T fixed. The relevance and divergence is enforced during training through the vector r.

We learn node and edge feature weights iteratively by recognizing and assigning weights to prominent node and edge features that satisfy queries of different types. Having all statistically driven computation of weights for edge features can minimize the side effect of poor node features and likewise computing weights for node features can decrease the consequences of poor edge features.

Algorithm 4 outlines the training procedure. I^+, I^- are the set of relevant and irrelevant categories for each document d in the ground truth that is used for training.

4.3.3 Query-time Inference

For a new document d,the inference problem is to choose k diversified categories. Using $H(d, \mathcal{C})$ we reduce the search space drastically and get the set S. Otherwise, we need to run our inference on entire set $\mathcal{C}$, which is very expensive. We then compute the node and edge feature matrices for all defined node and edge features. These individual feature matrices are then combined (using λ^T and W^T) to obtain vector b and matrix C. Based on Proposition 1, our inference objective is to choose k columns from the matrix $\tilde{C}$ such that their sum is as close as possible to b. Formally, the problem being solved is:

$$\underset{i_1...i_k}{argmin}\ D\left(b, \sum_{j=1}^{k} \tilde{C}^{i_j}\right) \tag{4.5}$$

where $i_1 ... i_k$ are indices of k columns of $\tilde{C}$.

Determining the exact solution (that is, $i_1 ... i_k$ columns) to the above optimization problem turns out to be computationally infeasible. Hence, we adopt an approximate solution based on submodular optimization. The choice of submodular optimization comes from the the fact that a greedy algorithm selecting an item in each iteration that maximizes the submodular objective is able to provide $1 - 1/e$ approximation guarantee. That is, the inferred categories from the greedy algorithm in the worst case is approximately 63% inferior to the optimum set of categories that an optimum algorithm can infer.

A set function f is said to be submodular if for any element v and sets $A \subseteq B \subseteq V \setminus \{v\}$, where V represents the ground set of elements, $\mathrm{f}(A \cup \{v\}) - \mathrm{f}(A) \geq \mathrm{f}(B \cup \{v\}) - \mathrm{f}(B)$. This is called the diminishing returns property and states, informally, that adding an element to a smaller set increases the function value more than adding that element to a larger set.

Our goal in Rel-Div inference is to find a subset $\mathcal{T}$ of k categories which are both relevant to the document and diversified among themselves. To this end, we propose the following optimization problem (we use the subscript i to indicate i^{th} element, that is, $b_i = b[i]$, $C_{i,j} = C[i,j]$):

$$\underset{|\mathcal{T}|=k}{\operatorname{argmax}} \quad \gamma \sum_{i \in \mathcal{T}} q_i b_i - \sum_{i,j \in \mathcal{T}} b_i C_{i,j} b_j \tag{4.6}$$

where γ is a positive regularization parameter that defines the trade-off between the two terms, and $\mathcal{T}$ consists of the indices of the k categories that will be returned in the ranking list. The vector $q = C \cdot b$. Intuitively, its i the element q_i measures the importance of c_i . To be specific, if c_i is similar to many categories (high $C_{i,j}$ $(j = 1, 2, ...,)$) that are relevant to the document (high b_j $(j = 1, 2, ...)$), it is more important than the categories whose neighbors are not relevant. For example, if c_i is close to the center of a big cluster relevant to the document, the value of q_i is large.

Intuitively, in the above maximization, the first term measures the weighted overall relevance of $\mathcal{T}$ with respect to the query, and q_i is the weight for c_i . It favors relevant categories from big clusters. In other words, if two categories are equally relevant to the

document, one from a big cluster and the other isolated, by using the weighted relevance, we prefer the former. The second term measures the similarity among the categories within $\mathcal{T}$. That is, it penalizes the selection of multiple relevant examples that are very similar to each other. By including this term in the objective function, we seek a set of categories which are relevant to the document, but also dissimilar to each other.

Algorithm 5 describes a greedy inference procedure. At each step we pick one column from $\tilde{C}$ that minimizes the distance in Equation 4.5 most. This is achieved by the first part $\left(\sum_{i\in\mathcal{T}} q_i b_i\right)$ of Equation 4.6 (maximizing the dot product ensures maximum similarity.) However, we also ensure that the picked column is most diverse from the already selected columns in the previous steps. This is achieved by the second part $\left(\sum_{i,j\in\mathcal{T}} b_i C_{i,j} b_j\right)$ of Equation 4.6. At the end of k steps we will have k diverse, but relevant documents.

Algorithm 4 Training

Input: Set of training data instances $\{d, I^+, I^-, N_f, C_f, r\}$
Output: $W^T$and λ^T
initialize variables $W^T$and λ^T
learn initial W^Tusing Logistic Regression ▷ uses $\{d, I^+, I^- N_f\}$
▷ $\tilde{C}, \tilde{C}_f$ used below are normalized C, C_f as in Proposition 1
while not converged($\mid b - \tilde{C}r \mid$) **do**
    b= compute relevance matrix using W^T and I^+
    find λ^T so that $D\left(b, \sum_f \lambda_f \tilde{C}_f r\right)$ is minimized
        ▷ W^T is fixed
    $p = \sum_f \lambda_f \tilde{C}_f r$
    find W^Tso that $D\left(\frac{1}{1+e^{-\sum_f w_f N_f}}, p\right)$ is minimized
        ▷ λ^T is fixed
end while
return $\left(W^T, \lambda^T\right)$

Algorithm 5 Inference

Input: Document d, Category catalog $\mathcal{C}$, λ^T, W^T, N_f,C_f

Output: k diverse interpretations

Generate $\mathcal{S} = H(q, \mathcal{C})$ and build a graph using documents in $\mathcal{S} = \{e_1, .., e_n\}$

Compute b using W^T and node features $N_{1..|N|}(q, \mathcal{S})$

Compute $C = \sum_f \lambda_f C_f(\mathcal{S})$ and normalize as in Proposition 1

$R = \{\emptyset\}$ ▷ set of selected indices

$Q = \{i_1, .., i_n\}$ ▷ indices to select

for $i = 1$ to k **do**

$\underset{c_k}{\text{argmax}} \quad \gamma \sum_{i \in \mathcal{T} \cup \{c_k\}} q_i b_i - \sum_{i,j \in \mathcal{T} \cup \{c_k\}} b_i C_{i,j} b_j$

$R = R \cup \{c_k\}$

end for

return k interpretations representing k columns $R_1, ..., R_{|R|}$

4.4 Experimental Evaluation

To validate our approach, we want to empirically evaluate the effect of diversity brought in by our technique. Hence, we evaluate our approach on diversity tasks from the literature, which focus on measuring the diversity factor in the result of a technique. To this end, we restrict our evaluation to the query/search result diversification task, which has been used in many works [48, 55] in the literature. Though the search result diversification is not the goal of our research (nevertheless our technique can used for the same task,) and validating our approach on this task provides us a few benefits: (i) These tasks are specifically designed for measuring the diversity aspect of the result, hence allowing us to validate and benchmark our approach on diversity (ii) The datasets introduced by these tasks are publicly accessible and human curated. That is, for every search query the diverse set of search results have been provided by the human judges (iii) We can use other methods that have reported their results on these datasets as baselines and compare our method against them.

In the subsequent sections we give an overview of the datasets that we use, our evaluation methodology and baselines compared.

4.4.1 Dataset

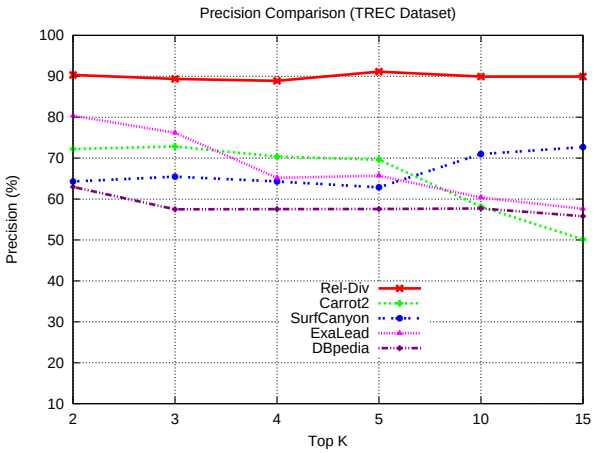
The QRU dataset used in SIGIR 2011 contains 100 TREC queries with diversified search results. These search results are formed by running the query against Google search engine and hand picking the diverse set of results by the human annotators. Since querying Google search engine through API is not open to the public and the Google search engine diversifies its own search results, we avoid using the Google search result to evaluate our algorithm. We restricted our space of query results to Wikipedia entities. We also experimented with ambiguous queries from the AMBIENT dataset which consists of 40 one word queries. These queries are in fact Wikipedia disambiguation page titles. This also made us to make use of Wikipedia articles as the corpus for querying.

4.4.2 Preprocessing

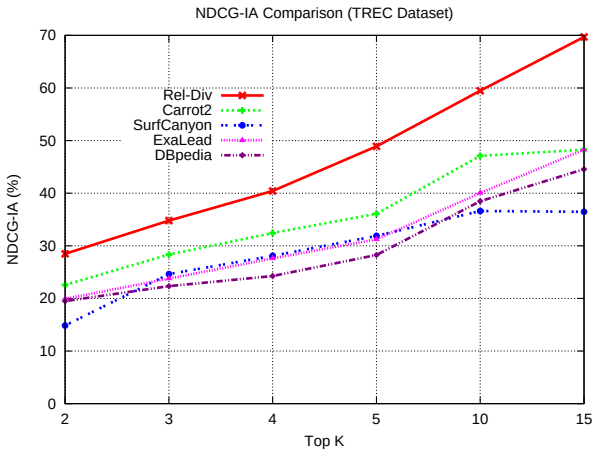
To validate our approach, we make use of the Wikipedia articles as corpus. Each Wikipedia article is associated with a text description created by the authors. We use this description to create a reverse index using Lucene [2]. The queries from QRU and AMBIENT datasets are used to retrieve the articles matching the query. We employed a keyword based Lucene search to query articles. We then created a complete graph of articles returned as the query results. Node features were defined to measure the overlap between the query terms and article terms. And, edge features were defined to measure the overlap (similarities) between a pair of articles (N-gram match, distance between the articles in Wikipedia category graph, and the like.)

To train and validate the system, we need *true* diversified results for the queries (QRU and AMBIENT.) To generate this ground truth, we employed 8 human annotators who initially ran the query against Wikipedia articles (using Lucene) and manually assigned a category for every query result. For example, for the query "Ambient", the categories assigned to the query result were "music", "album", 'physics', and the like. We limited the number of query results to 100 for each query. For each query, top 15 diversified results were identified by hand picking top search results spanning different categories.

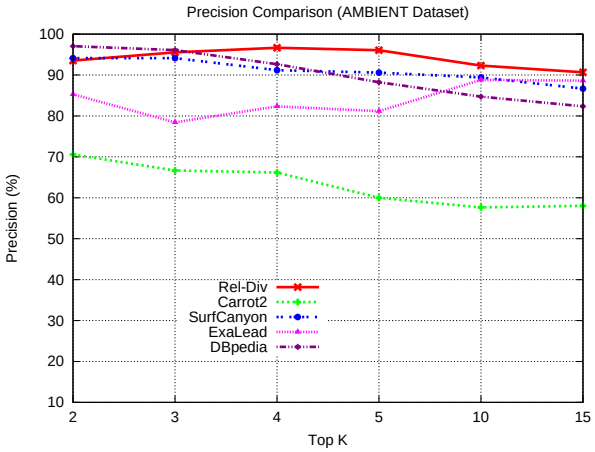
[2] `https://lucene.apache.org/`



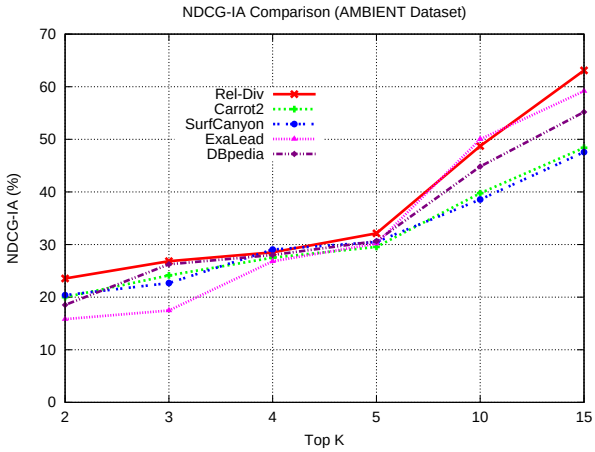
(a) Precision : TREC



(b) NDCG-IA : TREC



(c) precision : AMBIENT



(d) NDCG-IA : AMBIENT

Figure 4.2: Comparison with external systems

		Precision(%)			Recall(%)			NDCG-IA(%)		
		@5	@10	@10	@5	@10	@10	@5	@10	@10
TREC	Rel-Div	**91.13**	**89.93**	**89.83**	**7.02**	**13.85**	**20.4**	48.9	59.47	**69.7**
	M-Div	89.87	84.27	84.32	6.74	12.71	18.88	**49.71**	**62.68**	67.39
	M-Div-NI	83.75	80	80	6.83	12.81	19.35	42.48	60.88	66.52
	AFP	78.3	76.9	80.7	6.3	12.4	18.1	34.2	38.8	47.6
AMBIENT	Rel-Div	96.05	92.3	90.67	7.33	**14.57**	**21.61**	**32.12**	**48.72**	**63.1**
	M-Div	96.15	**94.15**	**93.56**	**7.43**	14.37	21.61	32.41	47.49	58.09
	M-Div-NI	**96.2**	93.58	93.19	7.33	13.87	21.11	22.93	43.59	55.84
	AFP	88.4	90.9	92.3	6.9	13.6	21.47	32.09	45.9	55.1

Table 4.1: Results of different approaches

4.4.3 Evaluation Methodology

The interpretations for each query are marked as relevant or irrelevant and each interpretation is assigned one or more categories. The system is trained on 30 percent of the queries and tested on the rest. We evaluated results on queries of length one or two. The relevance of any result to the query is measured using precision at different positions and the diversity is estimated using NDCG-IA [2]. Recall measurement is tricky since it. It is practically not possible to manually inspect all Wikipedia entities and determine how many are actually relevant for a query. Hence we based our recall on the candidate interpretations generated. We manually counted the number of relevant interpretations present in the candidate interpretations and measured how many of these relevant interpretations appeared in the top k interpretations.

In our experiments, we also consider a couple of other approaches to diversification, which have been reported in literature, though used in other problem settings. These include variants of GCD [48] and affinity propagation [55, 56].

- ***M-Div*** *:* Uses page rank matrix M as in GCD instead of the C_q matrix.
- ***M-Div-NI*** *:* Similar to M-Div, but node and edge weights are learned independently, without any iterations. This acts as GCD implementation.
- ***AFP****:Exemplar* nodes of Affinity propagation are taken as interpretations.

4.4.4 Comparison with Other Approaches

While experimenting with our proposed approach, we found best performance when the distance D in the div-step was chosen to be the KL-divergence and the D in the rel-step was chosen as the Euclidean distance. In Table 4.1, we compare the proposed diversification algorithm against M-Div, M-Div-NI and AFP on precision, recall and NDCG-IA measures.

We observed that our Ranking algorithm Rel-Div performs on par with (and sometimes even better than) M-Div and M-Div-NI. However, one of the major advantage of our

method compared to M-Div and M-Div-NI is that, we need not calculate the inverse of C_q matrix, which is a computationally intensive process for a large dimension matrices. We conclude from the results that the Rel-Div performs consistently better than other approaches when both relevance and diversification are considered across all types of queries.

4.4.5 Comparison Against Other Systems

We compare the diversity in search result using our approach against those from four other systems, *viz.*, carrot2, SurfCanyon, Exalead and DBPedia. The queries were directly fed to these system (through the respective web interface) and top 15 results were collected. With help of 8 human annotators, we assigned categories to each of the query results. Then the NDCG-IA metric is computed to compare the diversity aspect of these algorithms. We observed that the Rel-Div approach produces high diversity in the search results, which is evident from the Figure 4.2.

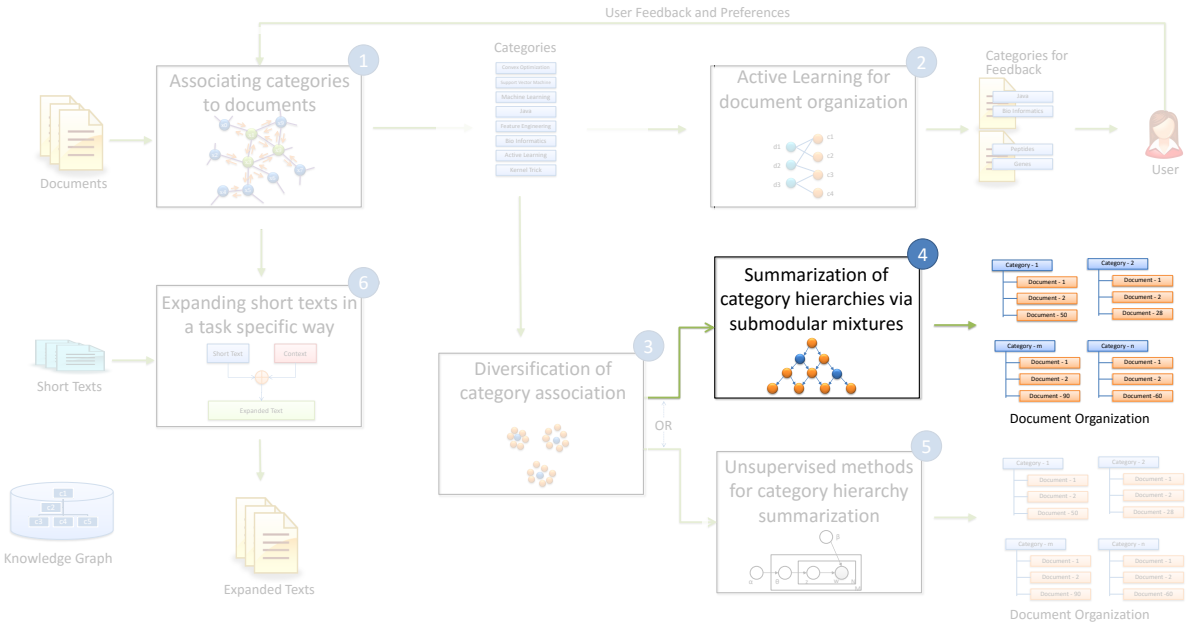
Note that, these external systems use their own corpus created by crawling the Web. Hence they are neither limited to Wikipedia article search nor they all have identical crawled corpus. Hence comparing these systems with Rel-Dev is tricky. Annotating each search result with a broad category (such as "music", "football", and the like) helps us neutralize the differences in their corpus and apply NDCG-IA metric as fairly as possible.

4.5 Conclusion

In this chapter we presented a body of techniques for generating top k diversified categories for a document using some internet encyclopedia, (in particular, Wikipedia was used in the experiments that were reported). Our approach is hinged on catering to two needs of the user, *viz.*, that all the categories are relevant and that they are as diverse as possible. We addressed this using a number of node features and edge features and learn these feature weights together interactively. We present experimental evaluations and find that our approach performs well on both the fronts (diversity and relevance) in comparison to existing techniques and publicly accessible systems.

Chapter 5

Category Summarization*



*__Published__: "Multi-Topic Summarization in DAG-Structured Topic Hierarchies via Submodular Mixtures", Ramakrishna Bairi, Ganesh Ramakrishnan, Rishabh Iyer, and Jeff Bilmes. In Proceedings of the Association for Computational Linguistics/Asian Federation of Natural Language Processing (ACL-IJCNLP), Beijing, China, 2015.

Chapter Summary

In this chapter we study the problem of summarizing DAG-structured category hierarchies over a given set of documents. The need for summarizing categories arises due to the large number of categories accumulated from all the documents in the collection. To get a smaller set of categories that are still representative of the documents in the collection, we need a sound summarization technique, which is the primary topic of this chapter. Other example applications of our proposed technique include automatically generating Wikipedia disambiguation pages for a set of articles, and generating candidate multi-labels for preparing machine learning datasets (for example, for text classification, functional genomics, and image classification). Unlike previous work, which focuses on clustering the set of documents using the category hierarchy as features, we directly pose the problem as a submodular optimization problem on a category hierarchy using the documents as features. Desirable properties of the chosen categories include document coverage, specificity, category diversity, and category homogeneity, each of which, we show, is naturally modeled by a submodular function. Other information, provided for example by unsupervised approaches such as LDA and its variants, can also be utilized by defining a submodular function that expresses coherence between the chosen categories and this information. We use a large-margin framework to learn convex mixtures over the set of submodular components. We empirically evaluate our method on the problem of automatically generating Wikipedia disambiguation pages using human generated clustering s as ground truth. We find that our framework improves upon several baselines according to a variety of standard evaluation metrics including the Jaccard Index, F1 score and NMI, and moreover, can be scaled to extremely large scale problems.

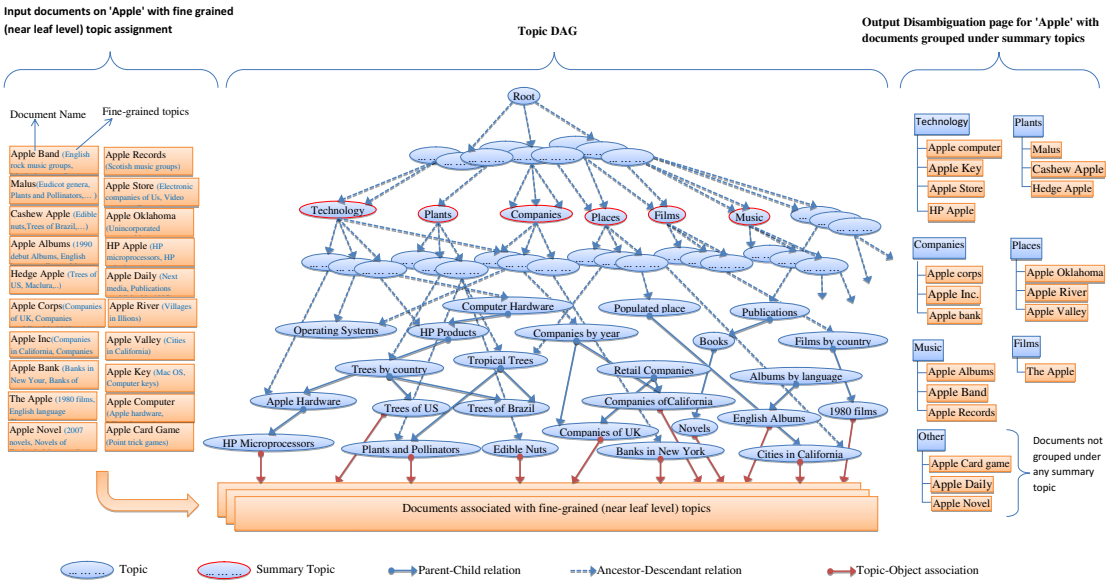


Figure 5.1: Category Summarization overview. On the left, we show many documents related to Apple. In the middle, a Wikipedia category hierarchy shown as a category DAG, links these documents at the leaf level. On the right, we show the output of our summarization process, which creates a set of summary categories (Plants, Technology, Companies, Films, Music and Places in this example) with the input documents classified under them.

5.1 Introduction

Several real world machine learning applications involve hierarchical categorization of topics for a set of objects. Objects could be, for example, a set of documents for text classification, a set of genes in functional genomics, or a set of images in computer vision. One can often define a natural category hierarchy to categorize these objects. For example, in text and image classification problems, each document or image is assigned a hierarchy of labels — a baseball page would be assigned the labels "baseball" and "sports." Moreover, many of these applications, naturally have an existing category hierarchy generated on the entire set of objects [118, 10, 160, 127, 143].

Given a DAG-structured category hierarchy and a subset of objects, we investigate the problem of finding a subset of DAG-structured categories that are induced by that subset (of objects). This problem arises naturally in several real world applications. For example, consider the problem of identifying appropriate label sets for a collection of articles.

Several existing text collection datasets such as the 20 Newsgroup[1], Reuters-21578[2] work with a predefined set of categories. We observe that these category names are overly general (too abstract)[3] for the articles categorized under them. On the other hand, techniques proposed in our previous chapters (Chapter 2, Chapter 3) and by systems such as Wikipedia Miner [101] and TAGME [53] generate several labels for each article in the dataset that are highly specific to the article. Collating all labels from all articles to create a label set for the dataset can result in a large number of labels and become unmanageable. Our proposed techniques can summarize such large sets of labels into a smaller and more meaningful label sets using a DAG-structured category hierarchy. This also holds for image classification problems and datasets like ImageNet [43]. We use the term *summarize* to highlight the fact that the smaller label set semantically covers the larger label set. For example, the categories *Physics*, *Chemistry*, and *Mathematics* can be summarized into a category *Science*.

A particularly important application of our work (and the one we use for our evaluations in Section 5.5) is the following: Given a collection of articles spanning different topics, but with similar titles, automatically generate a disambiguation page for those titles using the Wikipedia category hierarchy[4] as a category DAG. Disambiguation pages[5] on Wikipedia are used to resolve conflicts in article titles that occur when a title is naturally associated with multiple articles on distinct topics. Each disambiguation page organizes articles into several groups, where the articles in each group pertain only to a specific category. Disambiguation s may be seen as paths in a hierarchy leading to different articles that arguably could have the same title. For example, the title *Apple*[6] can refer to a plant, a company, a film, a television show, a place, a technology, an album, a record label, and a newspaper daily. The problem then, is to organize the articles into multiple groups where each group contains articles of similar nature (topics) and has an appropriately discerned group heading. Figure 5.1 describes the category summarization process for creation of the disambiguation page for "Apple".

[1] `http://qwone.com/~jason/20Newsgroups/`

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

[3] Category *Concept* is more abstract than the category *Science* which is more abstract than the category *Chemistry*

[4] `http://en.wikipedia.org/wiki/Help:Categories`

[5] `http://en.wikipedia.org/wiki/Wikipedia:Disambiguation`

[6] `http://en.wikipedia.org/wiki/Apple_(disambiguation)`

The choice of Wikipedia disambiguation page creation application to demonstrate our summarization technique is in line with the focus of this thesis – automatic categorization of documents. The disambiguation page can be thought of as an organization of various Wikipedia articles into different categories. Hence, automatic generation of Wikipedia disambiguation page is equivalent to automatic organization of digital documents. However, this particular choice of application gives us several benefits: (i) Wikipedia disambiguation pages are human curated categorization of articles and thus act as a sound dataset for the evaluation of our technique; (ii) The categories assigned to individual Wikipedia articles too are human curated. This reduces the noise in category assignment to the documents that gets introduced while applying techniques from Chapter 2, Chapter 3 or other systems such as TAGME and WikipediaMiner. Thus the error observed during our experiments reflects the error coming from the summarization technique, and is free from the errors introduced through document category association process.

All the above mentioned problems can be modeled as the problem of finding the most representative subset of category nodes from a DAG-Structured category hierarchy. We argue that many formulations of this problem are natural instances of submodular maximization, and provide a learning framework to create submodular mixtures to solve this problem. A set function $\mathrm{f}(.)$ is said to be submodular if for any element v and sets $A \subseteq B \subseteq V \setminus \{v\}$, where V represents the ground set of elements, $\mathrm{f}(A \cup \{v\}) - \mathrm{f}(A) \geq \mathrm{f}(B \cup \{v\}) - \mathrm{f}(B)$. This is called the diminishing returns property and states, informally, that adding an element to a smaller set increases the function value more than adding that element to a larger set. Submodular functions naturally model notions of coverage and diversity in applications, and therefore, a number of machine learning problems can be modeled as forms of submodular optimization [75, 78, 104, 68, 86, 88]. In this chapter, we investigate structured prediction methods for learning weighted mixtures of submodular functions to summarize categories for a collection of objects using DAG-structured category hierarchies. Throughout this chapter we use the terms "topic" and "category" interchangeably.

5.1.1 Related Work

To the best of our knowledge, the specific problem we consider here is new. Previous work on identifying categories can be broadly categorized into one of the following types:

a) cluster the objects and then identify names for the clusters; or b) dynamically identify topics (including hierarchical) for a set of objects. LDA [17] clusters the documents and simultaneously produces a set of topics into which the documents are clustered. In LDA, each document may be viewed as a mixture of various topics and the topic distribution is assumed to have a Dirichlet prior. LDA associates a group of high probability words to each identified topic. A name can be assigned to a topic by manually inspecting the words or using additional algorithms [96, 72]. LDA does not make use of existing topic hierarchies and correlation between topics. The Correlated Topic Model [16] induces a correlation structure between topics by using the logistic normal distribution instead of the Dirichlet. Another extension is the hierarchical LDA [15], where topics are joined together in a hierarchy by using the nested Chinese restaurant process. Nonparametric extensions of LDA include the Hierarchical Dirichlet Process [140] mixture model, which allows the number of topics to be unbounded and learned from data and the Nested Chinese Restaurant Process which allows topics to be arranged in a hierarchy whose structure is learned from data. In each of these approaches, unlike our proposed approach, an existing topic hierarchy is not used, nor is any additional object-topic information leveraged.

The pachinko allocation model (PAM)[85] captures arbitrary, nested, and possibly sparse correlations between topics using a DAG. The leaves of the DAG represent individual words in the vocabulary, while each interior node represents a correlation among its children, which may be words or other interior nodes (topics). PAM learns the probability distributions of words in a topic, subtopics in a topic, and topics in a document. We cannot, however, generate a subset of topics from a large existing topic DAG that can act as summary topics, using PAM.

HSLDA [107] introduces a hierarchically supervised LDA model to infer hierarchical labels for a document. It assumes an existing label hierarchy in the form of a tree. The model infers one or more labels such that, if a label l is inferred as relevant to a document, then all the labels from l to the root of the tree are also inferred as relevant to the document. Our approach differs from HSLDA since: (1) we use the label hierarchy to infer a set of labels for a *group* of documents; (2) we do not enforce the label hierarchy to be a tree as it can be a DAG; and (3) generalizing HSLDA to use a DAG structured hierarchy and infer labels for a *group* of documents (for example, combining into one big document) also

may not help in solving our problem. HSLDA will apply *all* the relevant labels to the documents as per the classifier that it learns for every label. Moreover, the "root" label is always applied and it is very likely that many labels near the top level of the label hierarchy are also classified as relevant to the group of documents.

Wei and James [14] present a hierarchical multi-label classification algorithm that can be used on both tree and DAG structured hierarchies. They formulate a search for the optimal consistent multi-label as the finding of the best sub graph in a tree/DAG. In our approach, we assume, individual documents are already associated with one or more topics and we find a consistent label set for a group of documents using the DAG structured category hierarchy.

Medelyan et al. [95] and Ferragina et al. [53] detect topics for a document using Wikipedia article names and category names as the category vocabulary. These systems are able to extract signals from a text document and identify Wikipedia articles and/or categories that optimally match the document and assign those article/category names as topics for the document. When run on a large collection of documents, these approaches generate enormous numbers of topics, a problem our proposed approach addresses.

5.1.2 Our Contributions

While most prior work discussed above focuses on the underlying set of documents, (*for example*, by clustering documents), we focus directly on the topics/categories. In particular, we formulate the problem as subset selection on the set of categories within a DAG while simultaneously considering the documents to be categorized. Our method can scale to the colossal size of the DAG (1 million categories and 3 million correlation links between categories in Wikipedia). Moreover, our approach can naturally incorporate outputs from many of the aforementioned algorithms. Our approach is based on submodular maximization and mixture learning, which has been successfully used in applications such as document summarization [87] and image summarization [142], but has never been applied to category identification tasks or, more generally, DAG summarization.

We introduce a family of submodular functions to identify an appropriate set of categories from a DAG structured hierarchy of categories for a group of documents. We character-

ize this category appropriateness through a set of desirable properties such as coverage, diversity, specificity, clarity, and relevance. Each of the submodular function components we consider are monotone, thereby ensuring a near optimal performance obtainable via a simple greedy algorithm for optimization.[7]. We also show how our technique naturally embodies outputs of other algorithms such as LDA, clustering, and classifications. Finally, we utilize a large margin formulation for learning mixtures of these submodular functions, and show how we can optimally learn them from training data.

Our approach demonstrates how to utilize the features collectively in the document space and the category space to infer a set of categories. From an empirical perspective, we introduce and evaluate our approach on a dataset of around 8000 disambiguation s that was extracted from Wikipedia and subsequently cleaned using the methods described in the experimentation section. We show that our learning framework outperforms many of the baselines, and is practical enough to be used on large corpora.

5.2 A Preliminary on Submodular Functions

Since this chapter heavily relies upon submodular functions and their properties, we give a brief introduction to the submodular functions in this section. The concepts presented in this section have been extensively studied in the literature, documented, and applied to many problems. Some of those concepts that are applicable to our work presented in the chapter have been reproduced in section for the completeness of reading.

5.2.1 Introduction

A submodular function is a set function. It has diminishing returns property. That is, the gain in the function value when an element is added to a subset is more than (or at the least equal to) the super-set. That is, as the set size increases, the increase in the function value by adding an additional element to it decreases.

[7] A simple greedy algorithm [105] obtains a $1-1/e$ approximation guarantee for monotone submodular function maximization

Let $f : 2^V \rightarrow \mathbb{R}_+$ be a set function and let $n = |V|$. The function f is called submodular if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, for all $A, B \subseteq V$.

f is called *monotone* if $f(A) \geq f(B)$ whenever $A \supseteq B$.

Equivalent definition (that can be deduced from the previous definition) is a follows: For any $A \subseteq B \subseteq V$ and $x \in V \setminus B$, $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$.

The above definition shows the property of diminishing returns mentioned earlier.

5.2.2 Operations Preserving Submodularity

In this section we define a few operations on submodular functions that preserver submodularity, and have been used in this chapter.

1. Non-negative weighted combinations of submodular functions are submodular. That is, if f_1,f_2,$\cdots$,f_k, are submodular and $\alpha_1, \alpha_2, \cdots, \alpha_k$ are non-negative numbers, the function $g(S) = \sum_{i=1}^{k} \alpha_i f_i(S)$ is submodular.

2. If f is a submodular function on V, and $T \subseteq V$, then the function defined by $g(S) = f(S \cap T)$ is also submodular.

3. If f is a submodular function on V, and $T \subseteq V$, then $g(S) = f(S \cup T)$ is submodular.

4. If f is monotonic, then the function $g(S) = \min(f(S), c)$, where c is a real number, is submodular.

5. If f is a submodular function on V, then $g(S) = f(V \setminus S)$ is also submodular.

6. Application of a concave function to a submodular function preserves submodularity. That is, if f is a submodular function on V, and ϕ is a concave function, then $g(S) = \phi(f(S))$, where $S \subseteq V$ is a submodular function.

5.2.3 Submodular Function Optimization

Submodular functions look similar to both convex and concave functions. From the definition of submodularity, these functions look more like concave functions (i.e., they have

non-increasing discrete derivatives). On the other hand, like convex functions, submodular functions are more convenient for minimization problems than maximization.

Submodular minimization problem is to find an optimal subset S that will minimize a submodular function f, i.e., $\arg\min_{S \subseteq V} f(S)$. This minimization without subject to any constraints is computable in polynomial time. However, Svitkina et al. [136] show that adding constraints like cardinality lower bound makes it NP hard and give approximation solutions with polynomial lower bounds.

On the other hand, maximization of submodular functions $\arg\max_{S \subseteq V} f(S)$ is NP-hard [105]. Greedy approximation algorithms are generally used to solve this problem in polynomial time with some approximation guarantees. In the following list we summarizes the approximation lower bounds that have been shown in the literature, and that are applicable to the kind of submodular functions used in this chapter.

- If the function is monotone, non-negative submodular with no constraints, Nemhauser et al. [105] show that a simple greedy algorithm gives $1 - 1/e$ approximation guarantee. They also show that the same approximation guarantee holds good with a cardinality constraint.
- If the function is non-monotone, symmetric submodular function, Feige et al. [51] show that a randomized set algorithm gives $1/2$ approximation in polynomial time.
- For unconstrained, arbitrary submodular functions Buchbinder et al. [24] give $1/2$ approximate polynomial time algorithm.

5.3 Problem Formulation

Let $G(V, E)$ be the DAG structured category hierarchy with V categories. These categories are observed to have a parent child (isa) relationship forming a DAG. Let D be the set of documents that are associated with one or more of these categories. The middle portion of Figure 5.1 depicts a category hierarchy with associated documents. The association links between the documents and categories can be hard or soft. In case of a hard link, a document is attached to a set of categories. Examples include multi-labeled

documents. In case of a soft link, a document is associated with a category with some degree of confidence (or probability). Furthermore, if a document is attached to a category t, we assume that all the ancestor categories of t are also relevant for that document. This assumption has been employed in earlier works [15, 14, 118] as well. Given a budget of K, our objective is to choose a set of K categories from V, which best describe the documents in D. The notion of best describing categories is characterized through a set of desirable properties – coverage, diversity, specificity, clarity, relevance and fidelity – that K categories have to satisfy. The submodular functions that we introduce in the next section ensure these properties are satisfied. Formally, we solve the following discrete optimization problem:

$$S^* \in \underset{S \subseteq V : |S| \leq K}{\operatorname{argmax}} \sum_i w_i f_i(S) \tag{5.1}$$

where, f_i are monotone submodular mixture components and $w_i \geq 0$ are the weights associated with those mixture components. Set S^* is the summary categories scored best.

It is easy to find massive (that is, size in the order of million) DAG structured categories hierarchies in practice. Wikipedia's category hierarchy consists of more than 1M categories (categories) arranged hierarchically. In fact, they form a cyclic graph [159]. However, we can convert it to a DAG by eliminating the cycles as described in the supplementary material. YAGO [133] and Freebase [19] are other instances of massive categories hierarchies. The association of the documents with the existing category hierarchy is also well studied. Systems such as WikipediaMiner [101], TAGME [53] and several annotation systems such as SegTag [46], Wikify [98], TaxonomyKernels [25] attach categories from Wikipedia (and other catalogs) to the documents by establishing the hard or soft links mentioned above.

Our goal is the following: Given a (ground set) collection V of categories organized in a pre-existing hierarchical DAG structure, and a collection D of documents, chose a size $K \in \mathbb{Z}_+$ representative subset of categories. Our approach is distinct from earlier work (for example, [71, 17]) where typically only a set of documents is classified and categorized in some way. We next provide a few definitions needed later in the chapter.

Definition 1: Transitive Cover Γ): A category t is said to cover a set of documents $\Gamma(t)$, called the transitive cover of the category t, if for all documents $i \in \Gamma(t)$, either i

is associated directly with category t or with any of the descendant categories of t in the category DAG. A natural extension of this definition to a set of categories T is defined as $\Gamma(T) = \cup_{t \in T} \Gamma(t)$.

Definition 2: Truncated Transitive Cover (Γ^{α})**:** This is a transitive cover of category t, but with the limitation that the path length between a document and the category t is not more than α. Hence, $|\Gamma^{\alpha}(t)| \leq |\Gamma(t)|$.

While our problem is closely related to clustering approaches, which consider the set of documents directly, there are some crucial differences. In particular, we focus on producing a clustering of documents where clusters are encouraged to honor a pre-defined DAG structured category hierarchy. Existing agglomerative clustering algorithms focusing on the coverage of documents may not produce the desired clustering. To understand this, consider six documents d1, d2 ... d6 to be grouped into three clusters. There may be multiple ways to do this depending upon multiple aggregation paths present in the category DAG: ((d1, d2), (d3, d4), (d5, d6)) or ((d1, d2, d3), (d4, d5), (d6)) or ((d1, d2, d3, d4), (d5), (d6)) or something else. Hence, we need more stringent measures to prefer one clustering over the others. Our work addresses this with a variety of quality criteria (coverage, diversity, specificity, clarity, relevance and fidelity, which are explained later in this chapter) that are organically derived from well established submodular functions. And, most importantly, we learn the right mixture of these qualities to be enforced from the data itself. Furthermore, our approach also generalizes these clustering approaches, since one of the components in our mixture of submodular functions is defined via these unsupervised approaches, and maps a given clustering to a set of categories in the hierarchy.

5.4 Submodular Components and Learning

Summarization is the task of extracting information from a source that is both small in size but still representative. Our problem is different from traditional summarization tasks since we have an underlying DAG as a category hierarchy that we wish to summarize in response to a subset of documents. Thus, a critical part of our problem is to take the graph structure into account while creating the summaries. Below, we identify properties

we wish our summaries to posses.

Coverage: A summary set of categories should cover most of the documents. A document is said to be covered by a category if there exists a path from the category, going through intermediary descendant categories, to the document, that is, the document is within the transitive cover of the category.

Diversity: Summaries should be as diverse as possible, that is, each summary category should cover a unique set of documents. When a document is covered by more than one category, that document is redundantly covered, for example, "Finance" and "Banking" would be unlikely members of the same summary.

Summary qualities also involve "quality" notions, including:

Specificity/Clarity/Relevance/Coherence: These quality measures help us choose a set of categories that are neither too abstract nor overly specific. They ensure that the categories are clear and relevant to the documents that they represent. When additional information such as clustering (from LDA or other sources) and tagging (manual) documents is available, these quality criteria encourage the chosen categories to show resemblance (coherence) to those clustering/tagging in terms of transitive cover of documents they produce.

In the section below, we define a variety of submodular functions that capture the above properties, and we then describe a large margin learning framework for learning convex mixtures of such components.

5.4.1 Submodular Components

We now investigate various classes of submodular functions (which we call, the submodular components)

5.4.1.1 Coverage Based Functions

Coverage components capture the "coverage" of a set of documents.

- **Weighted Set Cover Function:** Given a set of categories, $S \subseteq V$, define $\Gamma(S)$ as the set of documents covered — for each category $s \in S$, $\Gamma(s) \subseteq D$ represents the documents covered by category s and $\Gamma(S) = \cup_{s \in S} \Gamma(s)$. The weighted set cover function, defined as $f(S) = \sum_{d \in \Gamma(S)} w_d = w(\Gamma(S))$, assigns weights to the documents based on their relative importance (for example, in Wikipedia disambiguation, the different documents could be ranked based on their priority).

- **Feature-based Functions:** This class of function represents coverage in feature space. Given a set of categories $S \subseteq V$, and a set of features U, define $m_u(S)$ as the score associated with the set of categories S for feature $u \in U$. The feature set could represent, for example, the documents, in which case $m_u(S)$ represents the number of times document u is covered by the set S. U could also represent more complicated features. For example, in the context of Wikipedia disambiguation, U could represent TFIDF features over the documents. Feature based functions are then defined as $f(S) = \sum_{u \in U} \psi(m_u(S))$, where ψ is a concave (for example, the square root) function. This function class has been successfully used in several applications [77, 149, 150].

5.4.1.2 Similarity based Functions

Similarity functions are defined through a similarity matrix $\mathbf{S} = \{s_{ij}\}_{i,j \in V}$. Given categories $i, j \in V$, similarity s_{ij} in our case can be defined as $s_{ij} = |\Gamma(i) \cap \Gamma(j)|$, i.e the number of documents commonly covered by both i and j.

- **Facility Location:** The facility location function, defined as $f(S) = \sum_{i \in V} \max_{j \in S} s_{ij}$, is a natural model for k-medoids and exemplar based clustering, and has been used in several summarization problems [142, 149].

- **Penalty based diversity:** A similarity matrix may be used to express a form of coverage of a set S but that is then penalized with a redundancy term, as in the following difference: $f(S) = \sum_{i \in V, j \in S} s_{ij} - \lambda \sum_{i \in S} \sum_{j \in S,} s_{i,j}$ [89]). Here $\lambda \in [0, 1]$. This function is submodular, but is not in general monotone, and has been used in document summarization [89], as a dispersion function [21], and in image summarization [142].

5.4.1.3 Quality Control (QC) Functions

QC functions ensure a quality criteria is met by a set S of categories. We define the quality score of the set S as $F_q(S) = \sum_{s \in S} f_q(s)$, where $f_q(s)$ is the quality score of category s for quality q. Therefore, $F_q(S)$ is a modular function in S. We investigate three types of quality control functions: Category Specificity, Category Clarity, and Category Relevance.

- **Category Specificity:** The farther a category is from the root of the DAG, the more specific it becomes. Categories higher up in the hierarchy are abstract and less specific. We therefore prefer categories low in the DAG, but lower categories also have less coverage. We define $f_{\text{specificity}}(s) = s_h$ where s_h is the height of category s in the DAG. The root category has height zero and the "height" increases as we move down the DAG in Figure 5.1.

  For example, in Figure 5.2, $f_{\text{specificity}}(A)) = 0$, $f_{\text{specificity}}(B)) = 1$, and so on.

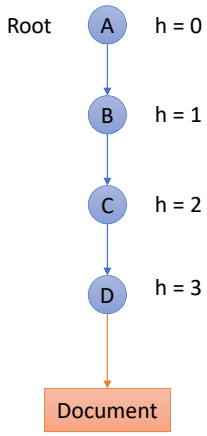


Figure 5.2: Specificity example

- **Category Clarity:** Category clarity is the fraction of descendant categories that cover one or more documents. If a category has many descendant categories that do not cover any documents, it has less clarity. Formally, $f_{\text{clarity}}(s) = \frac{\sum_{t \in \text{descendants}(s)} [\![\Gamma(t) > 0]\!]}{|\text{descendants}(s)|}$, where $[\![\cdot]\!]$ is the indicator function.

  For example, in Figure 5.3, $f_{\text{clarity}}(B) = 1$, because, all the descendants of category B are covering some of the documents. Meanwhile, $f_{\text{clarity}}(C) = 1/7 = 0.143$, since only one of the descendant category F is covering some documents.

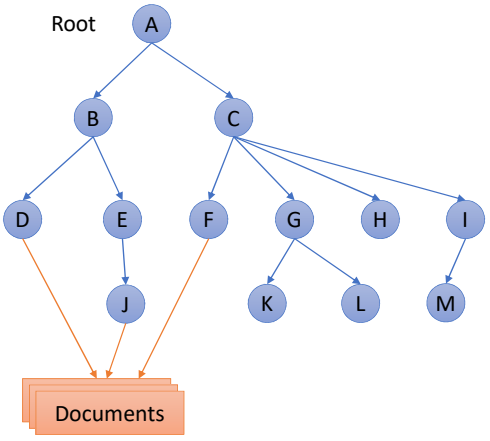


Figure 5.3: Clarity example

- **Category Relevance:** A category is considered to be more related to a document if the number of hops needed to reach the document from that category is lower. Given any set $A \subseteq D$ of document, and any category $s \in V$, we can define $f_{\text{relevance}}(s|A) = \arg\min_{\alpha}\{\alpha : A \subseteq \Gamma^{\alpha}(s)\}$.

  For example, in Figure 5.4, $f_{\text{relevance}}(B) = 2$, because, it takes 2 hops to reach a document from the category B. Meanwhile, $f_{\text{relevance}}(C) = 3$, since it takes 3 hops to reach a document from the category C.

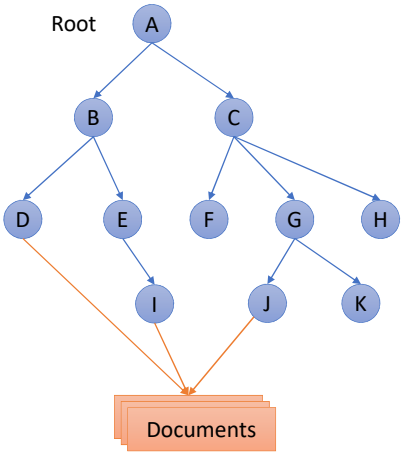


Figure 5.4: Relevance example

- **QC Functions As Barrier Modular Mixtures:** We introduce a modular

function for every QC function as follows

$$f^{\alpha}_{\text{specificity}}(s) = \begin{cases} 1 & \text{if the height of category } s \text{ is at least } \alpha \\ 0 & \text{otherwise} \end{cases}$$

for every possible value of α. This creates a submodular mixture with as many components as the number of possible values of α. In our experiments with Wikipedia, we had α varying from 1 to 120 stepping by 1, adding 120 modular mixture components. Similarly, we define,

$$f^{\beta}_{\text{clarity}}(s) = \begin{cases} 1 & \text{if the clarity of category } s \text{ is at least } \beta \\ 0 & \text{otherwise} \end{cases}$$

for every possible (discretized to make it countably finite) value of β. And,

$$f^{\gamma}_{\text{relevance}}(s) = f_{\text{cov}}(s|\Gamma^{\gamma}(s))$$

where $f_{\text{cov}}(.)$ is the coverage submodular function and $s|X$ indicates coverage of a category s over a set of documents X.

All these functions (modular and submodular terms) are added as mixture components in our learning framework to learn suitable weights for them. We then use these weights in our inference procedure to obtain a subset of categories as described in 5.4.2. We show from our experiments that this approach performs better than all other approaches and baselines.

5.4.1.4 Fidelity Functions

A function representing the fidelity of a set S to another reference set R is one that gets a large value when the set S represents the set R. Such a function scores inferred categories high when it resembles a reference set of categories and/or item clusters. The reference set in this case can be produced from other algorithms such as k-means, LDA and its variants or from a manually tagged corpus. Next we describe one such fidelity function.

- **Category Coherence:** This function scores a set of categories S high when the transitive cover (Definition 1) produced by the categories in S resembles the clusters

of documents produced by an external source (k-means, LDA or manual). Given an external source that clusters the documents, producing T clusters $L_1, L_2, ..., L_T$ (for T categories), category coherence is defined as:

$$f(S) = \sum_{t \in T} \max_{k \in S} w_{k,t}$$

where,

$$w_{k,t} = \left(\frac{1}{w^p_{k,t}} + \frac{1}{w^r_{k,t}} \right)^{-1}$$

$$w^p_{k,t} = \frac{|\Gamma(k) \cap L_t|}{|\Gamma(k)|}$$

$$w^r_{k,t} = \frac{|\Gamma(k) \cap L_t|}{|L_t|}$$

Note that, $w^p_{k,t} \geq 0$ and $w^r_{k,t} \geq 0$ are the precision or recall of the resemblance and $w_{k,t}$ is the F1 measure. If the transitive cover of categories in S resembles the reference clusters L_t exactly, we attain maximum coherence (or fidelity). As the resemblance diminishes, the score decreases. The above function $f(S)$ is monotone submodular.

5.4.1.5 Mixture of Submodular Components:

Given the different classes of submodular functions above, we construct our submodular scoring functions $F_w(\cdot)$ as a convex combinations of these different submodular functions $f_1, f_2, \ldots, f_m$, above. In other words,

$$F_w(S) = \sum_{i=1}^{m} w_i f_i(S), \tag{5.2}$$

where $w = (w_1, \ldots, w_m)$, $w_i \geq 0, \sum_i w_i = 1$. The components f_i are submodular and assumed to be normalized: that is, $f_i(\emptyset) = 0$, and $f_i(V) = 1$ for monotone functions and $\max_{A \subseteq V} f_i(A) \leq 1$ for non-monotone functions. A simple way to normalize a monotone submodular function is to define the component as $f_i(S)/f_i(V)$. This ensures that the components are *compatible* with each other. Obviously, the merit of the scoring function $F_w(\cdot)$ depends on the selection of the components.

5.4.2 Large Margin Learning

We optimize the weights w of the scoring function $F_w(\cdot)$ in a large-margin structured prediction framework. In this setting, we assume we have training data in the form of pairs of a set of documents, and a human generated summary as a set of categories. For example, in the case of Wikipedia disambiguation, we use the human generated disambiguation pages as the ground truth summary. We represent the set of ground-truth summaries as $\mathcal{S} = \{S_1, S_2, \cdots, S_N\}$. In large margin training, the weights are optimized such that ground-truth summaries $\mathcal{S}$ are separated from competitor summaries by a loss-dependent margin:

$$F_w(S) \geq F_w(S') + \mathcal{L}(S'), \quad \forall S \in \mathcal{S}, S' \in \mathcal{Y} \setminus \mathcal{S}, \tag{5.3}$$

where $\mathcal{L}(\cdot)$ is the loss function, and where $\mathcal{Y}$ is a structured output space (for example $\mathcal{Y}$ is the set of summaries that satisfy a certain budget B, that is, $\mathcal{Y} = \{S' \subseteq V : |S'| \leq B\}$). We assume the loss to be normalized, $0 \leq \mathcal{L}(S') \leq 1, \forall S' \subseteq V$, to ensure that mixture and loss are calibrated. Equation (5.3) can be stated as $F_w(S) \geq \max_{S' \in \mathcal{Y}} \left[F_w(S') + \mathcal{L}(S')\right], \forall S \in \mathcal{S}$ which is called *loss-augmented inference*. We introduce slack variables and minimize the regularized sum of slacks [86]:

$$\min_{w \geq 0, \|w\|_1 = 1} \quad \sum_{S \in \mathcal{S}} \left[\max_{S' \in \mathcal{Y}} \left[F_w(S') + \mathcal{L}(S')\right] - F_w(S)\right] + \frac{\lambda}{2}\|w\|_2^2, \tag{5.4}$$

where the non-negative orthant constraint, $w \geq 0$, ensures that the final mixture is submodular. Note a 2-norm regularizer is used on top of a 1-norm constraint $\|w\|_1 = 1$ which we interpret as a prior to encourage higher entropy, and thus more diverse mixture distributions. Tractability depends on the choice of the loss function. The parameters w are learned using stochastic gradient descent as presented by Tschiatschek et al. [142].

5.4.3 Loss Functions

A natural choice of loss functions for our case can be derived from cluster evaluation metrics. Every inferred category s induces a subset of documents, namely the transitive cover $\Gamma(s)$ of s. We compare these clusters with the clusters induced from the true categories in the training set and compute the loss.

In this chapter, we use the Jaccard Index (JI) as a loss function. Let S be the inferred categories and T be the true categories. The Jaccard loss is defined as

$$\mathcal{L}_{\text{jaccard}}(S,T) = 1 - \frac{1}{k}\sum_{s\in S}\max_{t\in T}\frac{|\Gamma(s)\cap\Gamma(t)|}{|\Gamma(s)\cup\Gamma(t)|}$$

where $k = |S| = |T|$ is the number of categories. When the clustering produced by the inferred and the true categories are similar, Jaccard loss is 0. When they are completely dissimilar, the loss is maximum, that is, 1. Jaccard loss is a modular function.

5.4.4 Inference Algorithm: Greedy

Having learned the weights for the mixture components, the resulting function $F_w(S) = \sum_{i=1}^{m} w_i f_i(S)$ is a submodular function. In the case when the individual components are themselves monotone (all our functions in fact are), $F_w(S)$ can be optimized by the accelerated greedy algorithm [102]. Thanks to submodularity, we can obtain near optimal solutions very efficiently. In case the functions are all monotone submodular, we can guarantee that the solution is within $1 - 1/e$ factor from the optimal solution [105].

The Algorithm 6 outlines the greedy inference procedure for submodular function optimization. It starts with an empty set A (step 3). At each step, the algorithm selects a category that gives the maximum marginal increase in the submodular function score (step 5). This category is then added to the set A (step 6). The marginal increase in the submodular score γ is computed (step 7) and compared against the threshold. If this increase is less than the threshold or if the size of set A has reached the budget (step 8), the algorithm terminates.

5.5 Experimental Results

To validate our approach, we make use of Wikipedia category structure as a category DAG and apply our technique to the task of automatic generation of Wikipedia disambiguation pages. The choice of Wikipedia disambiguation page creation application helps us to demonstrate our summarization technique, which is one of the key stages in handling the over-specified problem that is being addressed in this thesis. The disambiguation

Algorithm 6 Greedy Submodular Inference

Input : Wikipedia Category DAG $G(V, E)$as a BN, Document collection D with observed categories

Output : Set of categories

$A = \{\emptyset\}$

repeat

  $e = \underset{e \in V \setminus A}{argmax} \sum_i w_i f_i(e)$

  $A = A \cup \{e\}$

  $\gamma = \sum_i w_i f_i(A) - \sum_i w_i f_i(A \setminus \{e\})$

until $\gamma >$Threshold and $|A| <$Budget

return A

page can be thought of as an organization of various Wikipedia articles into different summary categories. Each Wikipedia article is associated with multiple categories by the editors of the articles. Taking all the categories from all the articles listed under a disambiguation page and adding them as disambiguation group titles will result in too many groups, which is equivalent to the over-specified problem in document organization described in Section 1.1.2. However, this particular choice of application gives us several benefits: (i) Wikipedia disambiguation pages are human curated categorization of articles and thus act as a sound dataset for the evaluation of our technique; (ii) The categories assigned to individual Wikipedia articles too are human curated. This reduces the noise in category assignment to the documents that gets introduced while applying techniques from Chapter 2, Chapter 3 or other systems such as TAGME and WikipediaMiner. Thus the error observed during our experiments reflects the error coming from the summarization technique, and is free from the errors introduced through document category association process.

We pre-processed the category graph to eliminate the cycles in order to make it a DAG. Each Wikipedia disambiguation page is manually created by Wikipedia editors by grouping a collection of Wikipedia articles into several groups. Each group is then assigned a name, which serves as a category for the group. Typically, a disambiguation page segregates around 20-30 articles into 5-6 groups. Our goal is to measure how accurately we can recreate the groups for a disambiguation page and label them, given only the collection

of articles mentioned in that disambiguation page (when actual groupings and labels are hidden).

5.5.1 Datasets

We parsed the contents of Wikipedia disambiguation pages and extracted disambiguation page names, article groups and group names. We collected approximately 8000 disambiguation pages that had at least four groups on them. Wikipedia category structure is used as the category DAG. We eliminated few administrative categories such as "Hidden Categories", "Articles needing cleanup", and the like. The final DAG had approximately 1M categories and 3M links.

5.5.2 Evaluation Metrics

Every group of articles on the Wikipedia disambiguation page is assigned a name by the editors. Unfortunately, these names may not correspond to the Wikipedia category names. For example, one of the groups on the "Matrix" disambiguation page has a name "Business and government" and there is no Wikipedia category by that name. However, the group names generated by our (and baseline) method are from the Wikipedia categories (which forms our category DAG). In addition, there can be multiple relevant names for a group. For example, a group on a disambiguation page may be called "Calculus", but an algorithm may rightly generate "Vector Calculus". Hence we cannot evaluate the accuracy of an algorithm just by matching the generated group names to those on the disambiguation page. To alleviate this problem, we adopt cluster-based evaluation metrics. We treat every group of articles generated by an algorithm under a category for a disambiguation page as a cluster of articles. These are considered as *inferred* clusters for a disambiguation page. We compare them against the actual grouping of articles on the Wikipedia disambiguation page by treating those groups as *true* clusters. We can now adopt Jaccard Index, F1-measure, and NMI (Normalized Mutual Information) based cluster evaluation metrics described in IR book [123]. For each disambiguation page in the test set, we compute every metric score and then average it over all the disambiguation pages.

5.5.3 Methods Compared

We validated our approach by comparing against several baselines described below. We also compared two variations of our approach as described next. For each of these cases (baselines and two variations) we generated and compared the metrics (Jaccard Index, F1-measure and NMI) as described in the previous section.

- **$\mathbf{KM_{docs}}$:** K-Means algorithm run on articles as TF-IDF vectors of words. The number of clusters K is set to the number of true clusters on the Wikipedia disambiguation page.
- **$\mathbf{KMed_{docs}}$:** K-Medoids algorithm with articles as TF-IDF vectors of words. The number of clusters are set as in $\mathrm{KM}_{\mathrm{docs}}$.
- **$\mathbf{KMed_{topics}}$:** K-Medoids run on categories as TF-IDF vectors of words. The words for each category is taken from the articles that are in the transitive cover of the category.
- **$\mathbf{LDA_{docs}}$:** LDA algorithm with the number of topics set to the number of true clusters on the Wikipedia disambiguation page. Each article is then grouped under the highest probability topic.
- **$\mathbf{SMML_{cov}}$:** This is the submodular mixture learning case explained in section 5.4.1.5. Here we consider a mixture of all the submodular functions governing coverage, diversity, fidelity and QC functions. However, we exclude the similarity based functions described in section 5.4.1.2. Coverage based functions have a time complexity of $O(n)$ whereas similarity based functions are $O(n^2)$. By excluding similarity based functions, we can compare the quality of the results with and without $O(n^2)$ functions. We learn the mixture weights from the training set and use them during inference on the test set to subset K categories through the submodular maximization (Equation 5.1).
- **$\mathbf{SMML_{cov+sim}}$:** This case is similar to $\mathrm{SMML}_{\mathrm{cov}}$ except that, we include similarity based submodular mixture components. This makes the inference time complexity $O(n^2)$.

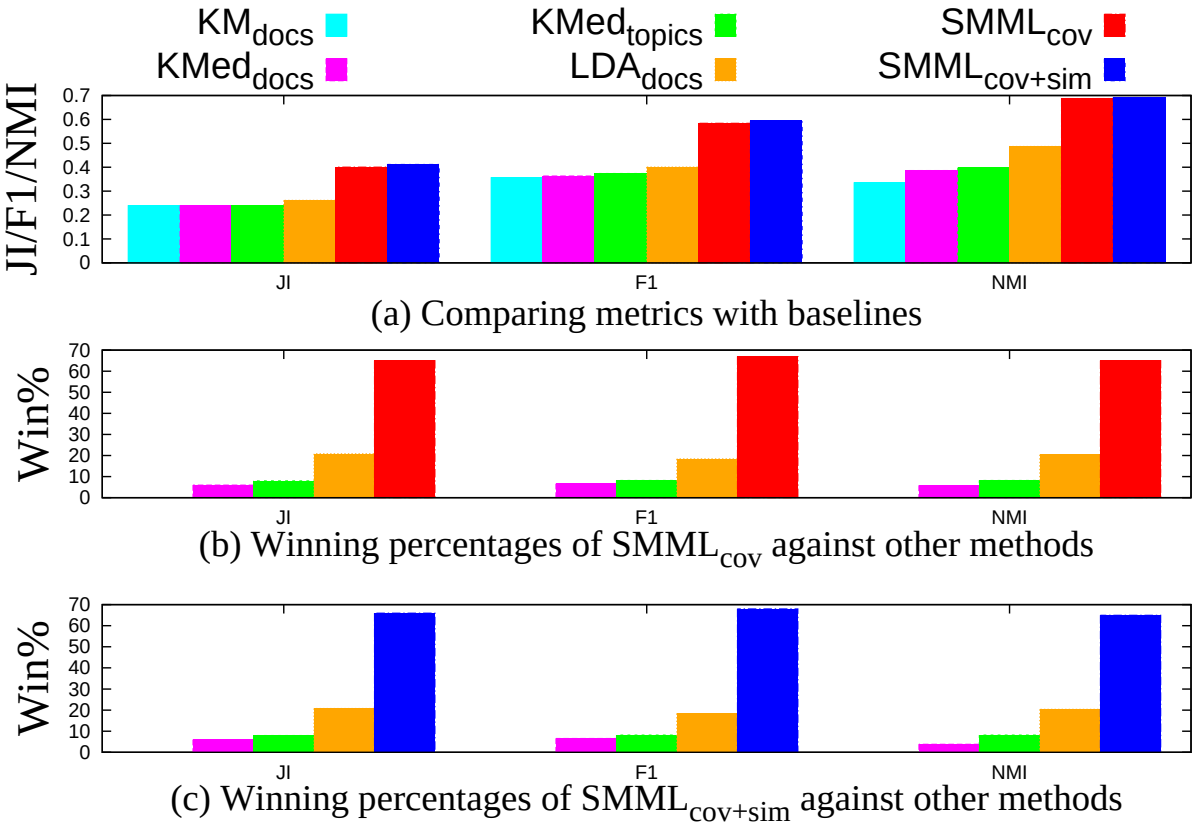


Figure 5.5: Comparison of techniques

We do not compare against HSLDA, PAM and few other techniques cited in the related work sections because they do not produce a subset of K summary topics, and hence they are not directly comparable with our work.

5.5.4 Evaluation Results

We show that the submodular mixture learning and maximization approaches, that is, SMML_{cov} and $\text{SMML}_{\text{cov+sim}}$ outperform other approaches in various metrics. In all these experiments, we performed 5 fold cross validation to learn the parameters from 80% of the disambiguation pages and evaluated on the rest of the 20%, in each fold.

In Figure 5.5a we summarize the results of the comparison of the methods mentioned above on Jaccard Index, F1 measure and NMI. Our proposed techniques SMML_{cov} and $\text{SMML}_{\text{cov+sim}}$ outperform other techniques consistently.

In Figures 5.5b and 5.5c we measure the number of test instances (that is, disambiguation queries) for where each of the algorithms dominate (win) for the evaluation metrics.

In 60% of the disambiguation queries, SMML_{cov} and $\text{SMML}_{\text{cov+sim}}$ approaches produce higher JI, F1 and NMI than all other methods. This indicates that the clusters of articles produced by our technique resembles the clusters of articles present on the disambiguation page better than other techniques.

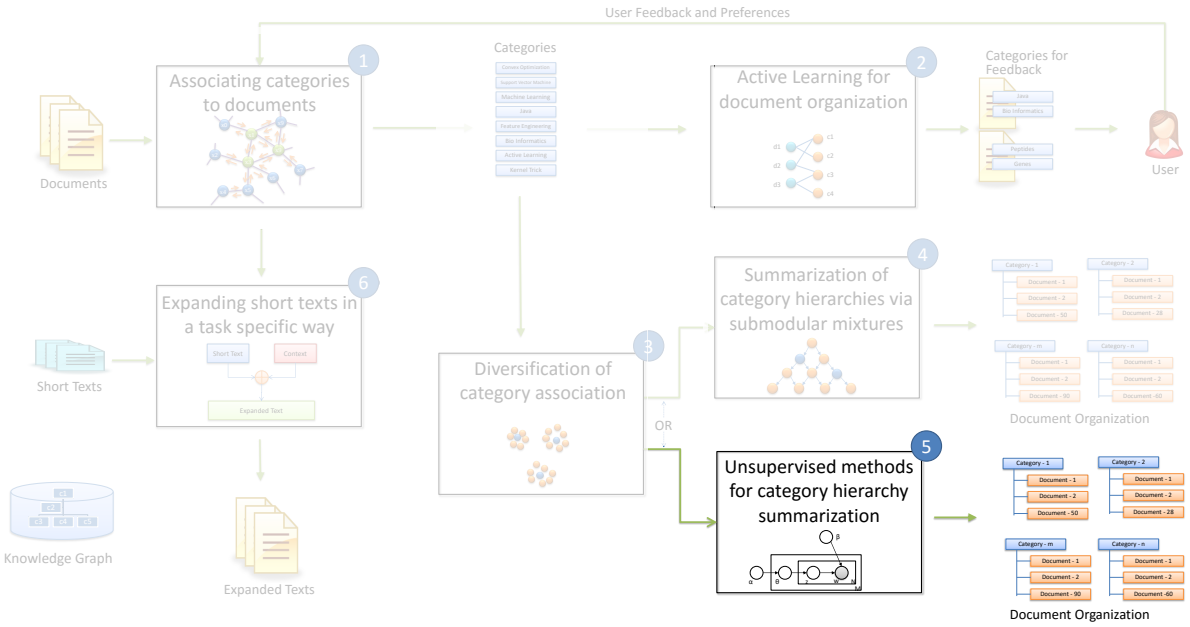
From Figures 5.5b and 5.5c it is clear that $O(n)$ time complexity based submodular mixture functions (SMML_{cov}) perform on par with $O(n^2)$ based functions ($\text{SMML}_{\text{cov+sim}}$), but at a greatly reduced execution time, demonstrating the sufficiency of $O(n)$ functions for our task. On the average, for each disambiguation query, SMML_{cov} took around 40 seconds (over 1M categories and 3M edges DAG) to infer the categories, whereas $\text{SMML}_{\text{cov+sim}}$ took around 35 minutes. Both these experiments were carried on a machine with 32 GB RAM, Eight-Core AMD Opteron(tm) Processor 2427.

5.6 Conclusions

We investigated a problem of summarizing categories over a massive category DAG such that the summary set of categories produced represents the objects in the collection. This representation is characterized through various classes of submodular (and monotone) functions that captured coverage, similarity, diversity, specificity, clarity, relevance and fidelity of the categories. Through this summarization we are able to reduce a large number of categories that get accumulated from all the documents in a document collection, thus addressing the "over-specified" categorization problem.

Chapter 6

Unsupervised Methods for Category DAG Creation*



*__Published__: "Beyond clustering: Sub-DAG Discovery for Categorizing Documents", Ramakrishna Bairi, Mark Carman, Ganesh Ramakrishnan, In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, Indianapolis, USA, Oct 24-28, 2016.

Chapter Summary

In this chapter we study the problem of generating DAG-structured category hierarchies, in an unsupervised setting, over a given set of documents associated with "importance" scores. In the previous chapter, we studied a technique for summarizing a set of categories through submodular mixture learning, which was a supervised technique. In this chapter we present a few unsupervised techniques for generating a DAG-structured category hierarchy to address the "over-specified" problem of document categorization. As in the previous chapter, we can use this technique for automatically generating Wikipedia disambiguation pages for a set of articles having click counts associated with them. Here we pose the problem as that of finding a DAG structured generative model that has maximum likelihood of generating the observed "importance"scores for each documents where document are modelled as the leaf nodes in the DAG structure. Desirable properties of the categories in the inferred DAG-structured hierarchy include document coverage and category relevance, each of which, we show, is naturally modeled by our generative model. We propose two different algorithms for estimating the model parameters. One by modeling the DAG as a Bayesian Network and estimating its parameters via Gibbs Sampling; and the other by estimating the path probabilities using the Expectation Maximization algorithm. As in previous chapter, we empirically evaluate our method on the problem of automatically generating Wikipedia disambiguation pages using human generated clustering s as the ground truth. We find that our framework improves upon the baselines according to the F1 score and Entropy that are used as standard metrics to evaluate the hierarchical clustering.

6.1 Introduction

With the exponential growth of digital artifacts (such as texts, images, and the like,) particularly on the Web, hierarchical organization of these artifacts is becoming increasingly important to manage the data. Many machine learning applications in several domains such as document processing, functional genomics, image processing, etc., deal with hierarchical organization of documents, genes, images, etc., in their respective domains. The underlying hierarchical structure identifies the relationships of dependence between different categories and provides valuable sources of information for categorization. For example, in document classification problems, a baseball document would be assigned the hierarchy of labels – "baseball" , "team sports" and "sports." Although considerable research has been conducted in the field of hierarchical document categorization [118, 10, 160, 127, 143], little has been done on automatic organization of artifacts (documents) using a Knowledge Graph.

Many techniques have been proposed for category hierarchy creation for organizing digital artifacts. While supervised techniques [49, 134, 60, 42] try to learn the hierarchy from the labeled examples, unsupervised methods [113, 107, 15] try to infer the hierarchy from the data (artifacts) itself, without recourse to human generated labels. Another class of research [122, 67] has been actively looking into adapting a gen etc "global" category hierarchy (such as Wikipedia/Freebase/DBPedia concept hierarchy) into a specific "local" categorization. These methods are able to associate the artifacts (text documents) with the closely matching concept nodes from the "global" category hierarchy. The matched concept nodes thus become categories for the artifacts. These methods not only associate curated category names from the "global" category hierarchy, but also bring in rich semantics for the categories from the "global" hierarchy. This motivates us to focus on this class of approaches for category hierarchy generation in our current work. It is easy to find massive (that is, size in the order of millions) DAG structured category hierarchies in practice. Wikipedia's category hierarchy consists of more than 1.5M categories (categories) arranged hierarchically. YAGO [133] and Freebase [19] are other instances of "global" category hierarchies.

It is very common to have some sort of "importance" weights/scores associated with the

artifacts in a collection. For example, in a web page collection, each web page may have a click count (the number of times the page has been viewed or accessed) associated with it. In Hierarchical tag visualization [129] applications, leaf tags may have document counts (number of documents assigned to the tag) associated with them. And, in an advertisement collections, each advertisement may have a revenue or cost associated with it. The existing methods for category identification [49, 134, 60, 42, 113, 107, 122, 67] do not focus on the hierarchy generation from the artifact importances. To reflect the artifact importance in the choices of category nodes of the hierarchy, we propose novel approaches to hierarchy generation. Specifically, we propose a generative model for explaining the observed artifact importances based on the hierarchy structure and apply Gibbs sampling and EM methods to estimate the parameter of these models.

As in the last chapter, we focus on the following problem: Given a DAG-structured category hierarchy and a collection of artifacts with associated importances, we investigate the problem of finding a sub-DAG of DAG-structured categories that are induced by the artifacts. This problem is motivated from the following real world applications (repeated here from the last chapter for the continuity of reading):

- *Automatic generation of hierarchical disambiguation page for Wikipedia*: Given a collection of articles spanning different categories but with similar titles, automatically generate a hierarchical disambiguation page for those titles using the Wikipedia category hierarchy[1]. This problem is explained in detailed in the last chapter, and is also used in our evaluations in 6.6. However, there is one major addition in the current chapter – "click count". Each article in Wikipedia has an associated click count, which tells how many times that article has been opened and read. Currently, Wikipedia does not consider this while grouping articles on a Disambiguation page, since Disambiguation pages are created manually. Our proposed approach is able to leverage this information while generating the hierarchical groups automatically. Figure 6.1 describes the process of category DAG creation for the disambiguation page for the term "Apple".

- *Handling over-specified categorization*: Techniques proposed by systems such as Wikipedia Miner [101] and TAGME [53] generate several labels for each article in

[1] `http://en.wikipedia.org/wiki/Help:Categories`

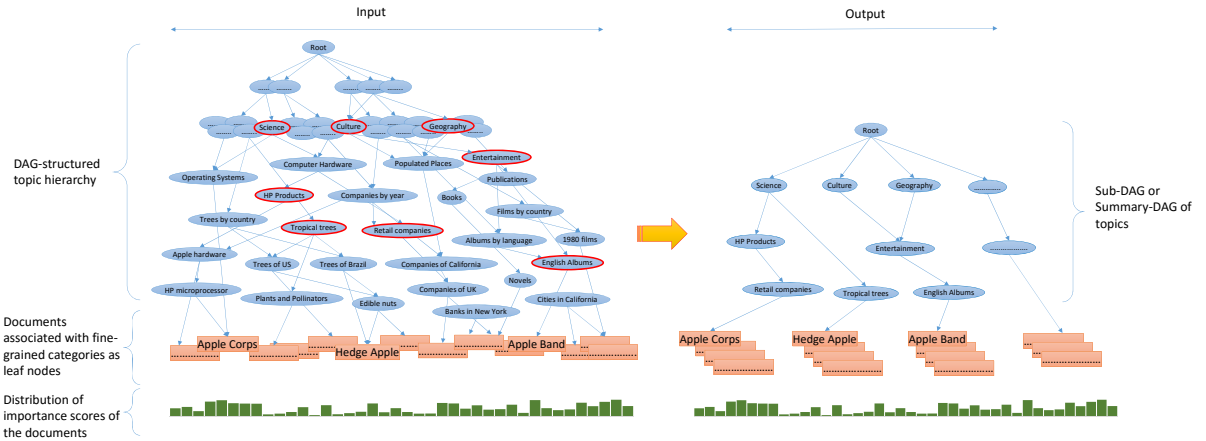


Figure 6.1: Overview of sub-DAG selection: Documents with the importance scores associated as leaf nodes of DAG-structured category hierarchy is given as input to our method. We generate a sub-DAG from the DAG-structured category hierarchy such that the documents generated from this sub-DAG (using the estimated parameters) will have the distribution of importance scores as close as possible to the observed distribution.

> the dataset from the Wikipedia (pages and categories) and are highly specific to the article. Collating all labels from all articles to create a label set for the dataset can result in a large number of labels and become unmanageable. We need a hierarchy of these labels to manage the dataset better. Our proposed techniques can discover a suitable label hierarchy (as a sub-DAG) from such large sets of labels using a "global" DAG-structured category hierarchy (such as Wikipedia).

We model the above problems as that of finding the most representative sub-DAG of category nodes from a DAG-Structured category hierarchy. We model this as a two step process. During the first step, we build a generative model that is able to produce the observed importance scores for the artifacts from the DAG-Structured category hierarchy. In this process, certain category nodes in the DAG-Structured category hierarchy become more important than other nodes for generating the observed importance scores for the artifacts. In the second step, we collect those important nodes and the edges that connect them (possibly indirectly) from the DAG-Structured category hierarchy to produce a sub-DAG. The first step is akin to high recall step, where the entire DAG-Structured category hierarchy is used to generate the importance scores. Meanwhile, the second step is akin to the high precision step where a few nodes that contribute maximally to the generation

of importance scores are selected.

6.1.1 Related Work

Previous works on hierarchical organization can be broadly classified into two groups: (i) hierarchical clustering and (ii) hierarchical topic modeling

The hierarchical clustering based algorithms [161] groups the documents at different levels. At the lower most level, every document is in its own group and at the top level, all documents are in one groups. To form the next level, most similar groups in the current level are merged.

The specific problem we consider here is different from hierarchical clustering. In our problem, we try to generate a hierarchy of labels as a sub-DAG from an existing DAG that can best describe the observed documents. This problem has several differences from hierarchical clustering: (i) The clustering uses category hierarchy as the features than the documents. (ii) It can naturally associate the labels to the clusters from the category hierarchy. (iii) a sub-DAG of label space (category) along with the documents categorized under them are output. (iv) the sub-DAG of categories identified by our algorithm likely to generate the observed counts/scores at the leaf level.

Topic model based techniques group high probability words in each identified topic. Topics can form a hierarchy in which topics at top levels group very abstract words compared to the topics at lower levels. Notable algorithms in hierarchical topic models include Hierarchical LDA [15], Pachinko Allocation Model (PAM)[85], HSLDA [107].

Hierarchical LDA [15] presents the nested Chinese restaurant process (nCRP), a stochastic process that assigns probability distributions to ensembles of infinitely deep, infinitely branching trees of topics. Here, documents are modeled as paths down a random tree, and the preferential attachment dynamics of the nCRP leads to clustering of documents according to sharing of topics at multiple levels of abstraction. Hierarchical Dirichlet Process [140] is a nonparametric generalization of Latent Dirichlet Allocation (LDA), where the number of topics can be unbounded and learned from data. In these approaches, unlike our proposed approach, an existing topic hierarchy is not used, and the existing artifact-topic information is not leveraged.

The Pachinko Allocation Model (PAM)[85] models documents as a mixture of distributions over a single set of topics, using a directed acyclic graph to represent topic co-occurrences. Each node in the graph is a Dirichlet distribution. At the top level there is a single node. Besides the bottom level, each node represents a distribution over nodes in the next lower level. The distributions at the bottom level represent distributions over words in the vocabulary. However, the difficulty is in generating a subset of topics from a large existing topic DAG that can act as summary topics due to the presence of too many nodes leading to prohibitively long execution time.

Perotte et al. proposed a hierarchically-supervised topic model (HSLDA) [107] combining LDA model with the classification algorithm. They make use of hierarchical structure of ICD (International Classification of Diseases) codes during learning step. The hierarchy is assumed to be in the form of a tree. During the training phase, constraints they impose constraints so that the predicted code forms a path from the root to the leaf. Our approach differs from HSLDA since: (1) we use the label hierarchy to infer a set of labels for a *group* of documents; (2) we do not enforce the label hierarchy to be a tree as it can be a DAG; and (3) we do not have labeled data (We use the ground truth from the Wikipedia disambiguation dataset only for evaluating the algorithm, and not for training the model.)

6.1.2 Our Contributions

Our approach is based on generative models which have been successfully used in applications such as document modeling [151], but have to the best of our knowledge never been applied to sub setting a category DAG to create a sub-DAG describing the document collection. By modeling the DAG-structured category hierarchy as a Markov network, we introduce a procedure to estimate the marginal probabilities of the nodes in generating the observed grouping of the documents along with their importance scores [2]. Our approach is based on Gibbs sampling with *path* constraints to ensure root-to-leaf path for every document is maintained. Unlike other methods [107, 17, 85, 151] we do not observe the words in the documents. The co-occurrence statistics of the leaf level categories in

[2]if documents don't have importance scores associated with them then their importance is simply set to be the same

the documents drive the sampler to sample common ancestors more frequently, thus increasing the marginal probabilities of those nodes. While the path sampling approaches [15, 73] fail in the case of massive DAG-structured hierarchies (such as Wikipedia category hierarchy,) due to extremely large number of paths, our approach is able to do local sampling (within the Markov Blanket). We also present an EM based method to estimate the expected importance of every edge in the DAG-structured category hierarchy based on the importance scores of the documents. Using the expected importance of edges in the DAG we propose a technique for a sub DAG generating. From an empirical perspective, we introduce and evaluate our approach on a dataset of around 800 disambiguation s that was extracted from Wikipedia and subsequently cleaned using the methods described in the experimentation section. We show that our method outperforms other baselines, and is practical enough to be used on large corpora.

6.2 Problem Formulation

Let $G(V; E)$ be the DAG structured category hierarchy with V categories. As in the last chapter (Chapter 5), we assume these categories to have a parent child (is-a) relationship, forming a DAG. Let D be the set of documents that are associated with one or more of these categories. The left part of Figure 6.1 depicts a category hierarchy with associated documents. If a document is attached to a category t, we assume that all the ancestor categories of t are also relevant for that document. This assumption has been employed in earlier works [15, 14, 118] as well. Furthermore, we assume that there exists a function associating importance scores with every document node. Examples of scores can be click counts of the documents, number of likes given to the document, and the like. Given a budget of K, our objective is to choose a DAG of K categories from $G(V; E)$, which best describes the documents in D. The notion of best describing categories is characterized through a generative process which can generate the observed importance scores at the document nodes.

In this work we expect the user to input the value for K. One possible way to choose the number of categories is by visually inspecting the hierarchy formed and ensuring that properties of hierarchy (such as tree width, depth, sparsity, etc.) are within the acceptable

limits. But soon this can become unmanageable for larger datasets. To get some hint about K, we can plot the Entropy or F1 score for various values of K as shown in Figure 6.4 in our experimental section, and choose the value of K at which the curve starts plateauing.

As in the last chapter (Chapter 5), we make use of Wikipedia's category hierarchy as a massive DAG. It consists of more than 1M categories (categories) arranged hierarchically. In fact, they form a cyclic graph [159]. However, we can convert the graph to a DAG by eliminating the cycles [3]. Systems such as WikipediaMiner [101], TAGME [53] and several annotation systems such as [98, 25] attach categories from Wikipedia (and other catalogs) to the documents.

Our goal is the following: Given,

1. A DAG-structured category hierarchy $G(V; E)$ associated with the categories and documents, over $V = V_{\text{categories}} \cup V_{\text{docs}}$ nodes, which contains $\left|V_{\text{categories}}\right|$ internal nodes (categories) denoted: $c_1, ..., c_{\left|V_{\text{categories}}\right|}$ and $\left|V_{\text{categories}}\right|$ leaf nodes (documents) denoted: $d_1, ..., d_{\left|V_{\text{categories}}\right|}$
2. A function associating scores with each of the document nodes: $Count(d_i) \in \mathbf{N}_+$. Note, we assume without loss of generality, the scores to be positive integers. Any set of positive real number scores can be appropriately scaled and rounded to produce integer scores.

Estimate a model that can predict the observed scores, that is:

1. Associate a Bayesian Network over binary variables with the structure given by the DAG above, (that is, let X_i be a binary variable corresponding to node c_i and Y_j be a binary variable corresponding to node d_j).
2. Determine the parameters of a Bayesian Network, that is, $P(X_i|parents(X_i))$ and $P(Y_j|parents(Y_j))$ such that the leaf nodes have marginal probability proportional to their scores, that is, $P(Y_j = 1) = Count(d_j)/\sum_k Count(d_k)$.

[3] our cycle detection and elimination algorithm is described in Appendix A

3. Construct a sub-DAG of K nodes by first selecting the K nodes from the Bayesian Network that have maximum marginal probability and entropy over its children and then interconnecting them with the edges from the DAG. These nodes contribute most in predicting the observed score.

In order to train the network we generate training data assenting the leaf variables (that is, documents) don't co-occur and have frequency given by their counts. That is, we generate example data for learning the network as unit vectors of the form: – $(Y_1, ..., Y_N) = (0, ..., 0, 1, 0, ...)$ – where $P(Y_i = 1) = Count(d_i)/\sum_j Count(d_j)$

6.3 Gibbs Sampling based Parameter Estimation

Given a DAG structured category hierarchy $G = (V, E)$ and a set of documents (with importance scores) attached as the leaf nodes in the category DAG, we assume a Bayesian Network (BN) with category nodes. Our goal is to first estimate the parameters of this BN such that, a generative process using this BN to assign to the binary leaf nodes (documents) is able to produce the desired marginal distribution given by the observed document scores. With this we mean that, repeated sampling of nodes from this BN (with the estimated parameters) is able to produce the leaf node (document) the number of times proportional to its importance score. After estimating the parameters of the BN, in the next step, we produce a ranking of the category nodes using their marginal probabilities and entropy over their children to construct a sub-DAG of K nodes with approximately the same marginal distribution over the leaves. This step is described in Section 6.5. In this section, we focus on estimating the parameters of the BN.

Let $X = (X_v)_{v \in V}$ be a set of random variables indexed by the category nodes V. For a BN with respect to G, its joint probability density function can be written as a product of the individual density functions, conditioned on the parent variables:

$$p(x) = \prod_{v \in V} p\left(x_v \,\middle|\, x_{\pi(v)}\right) \tag{6.1}$$

where $\pi(v)$ is the set of parents of v.

Here $p(x)$ is the probability of observing a particular assignment of categories to a document. Specifically,

$$p(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} p\left(X_v = x_v \mid \forall_{j \in \pi(v)}\ X_j = x_j\right) \tag{6.2}$$

All the observations x_i are binary, taking values 0 or 1. If $x_i = 1$, the i^{th} category is assigned to the document; otherwise, it is not assigned to the document. Hence, there will be $2^{\pi(i)}$ number of parent configurations for any category X_i.

Once the parameter θ of this BN, that is, $p\left(x_v \,\middle|\, x_{\pi(v)}\right)$ for all the categories in the G has been estimated, the likelihood of observing the document collection D from this BN can be computed as:

$$LL(X|\theta) = \sum_{d \in D} \sum_{v \in V} \log\left(p\left(X_v = x_v \,\middle|\, X_{\pi(v)} = x_{\pi(v)}\right)\right) \tag{6.3}$$

Given a assignment of valves to the variables in a BN, a Gibbs sampler simply iterates over each latent categoryX_i(note that there is one BN for each document) sampling a new value for the variable according to its posterior distribution:

$$X_i \sim \text{Bernoulli}(P), \text{ where} \tag{6.4}$$

$$P = p(X_i = 1 \mid X_{-i}) \tag{6.5}$$

$$1 - P = p(X_i = 0 \mid X_{-i}) \tag{6.6}$$

Here X_{-i} denotes all nodes in the BN except for X_i.

The conditional independence property of a BN states that any two nodes (u, v) are conditionally independent given a set of variables z which d-separates them:

$$X_u \perp\!\!\!\perp X_v \,|\, X_Z \tag{6.7}$$

The Markov Blanket of node v, denoted $\mathcal{MB}(v)$, is the minimal set of nodes which d-separates node v from all other nodes. Using this property, Equation 6.5 and 6.6 simplify to:

$$P = p\left(X_i = 1 \mid X_{\mathcal{MB}(i)}\right) \tag{6.8}$$

$$1 - P = p\left(X_i = 0 \mid X_{\mathcal{MB}(i)}\right) \tag{6.9}$$

Furthermore, the conditional distribution of one variable given all others is proportional to the joint distribution:

$$p\left(x_j|x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n\right) \propto p\left(x_1,\ldots,x_n\right)$$

"Proportional to" in this case means that the denominator is not a function of x_j and thus is the same for all values of x_j; it forms part of the normalization constant for the distribution over x_j.

Applying this principle to Equation 6.8, we have

$$p\left(X_i = 1 \mid X_{-i}\right) \propto p\left(X_i, X_{\mathcal{MB}(i)}\right) \tag{6.10}$$

Using the factorization stated in Equation 6.1 and 6.2, we have

$$\begin{aligned} p\left(X_i{=}1 \mid X_{-i}\right) &\propto p\left(X_i{=}1, X_{\mathcal{MB}(i)}\right) \\ &\propto p\left(X_i{=}1 \,|\, X_{\pi(i)}\right) \prod_{k \in \text{children}(i)} p\left(X_k \,|\, X_{\pi(k)}\right) \end{aligned} \tag{6.11}$$

Similarly, $p\left(X_i = 0 \mid X_{-i}\right)$ is computed. Note that, the normalization constant is computed by samples summing the right side of Equation 6.11 for $X_i = 1$ and $X_i = 0$.

The Gibbs Sampler samples repeatedly from the posterior in Equation 6.11. When the Gibbs Sampler reaches a steady state, we would have estimated the parameters of the BN, that is, $p\left(x_v \,\middle|\, x_{\pi(v)}\right)$ for all $v \in V$.

The Algorithm 7 describes the Gibbs Sampling. The logarithm interactively samples the categories for the documents as per the posterior distribution in Equation 6.11. In order to reflect the importance scores of the documents in the posterior estimation, we create $M = \sum_d \text{count}(d)$instances of the BN one for each of the training examples.Similar to the collapsed Gibbs sampler for the LDA [131] we create counters to hold the number of times a category is sampled for a document (lines 7-13). Repeatedly sampling the latent categories for the documents as per the estimated posterior (from the counter values

accumulated so far) and then updating the counters based on the samples observed, makes the estimated posterior converge to its true posterior (lines 15-32.) To ensure root to leaf path in the categories sampled for a document, we enforce two constraints during the sampling (line 28):

- Set $X_{di} = 1$ if there exist a child X_k of X_i such that $X_{dk} = 1$, but all of its parents are set to 0, that is, $X_{d\pi(k)} = 0$

  This constraint ensures that the parent category is set to 1 if there is a child whose all parents are set to 0.

- Set $X_{di} = 0$ if there does not exist a child X_k of X_i such that $X_{dk} = 1$

  This constraint ensures that, parent is set to 0, if none of its children are set to 1

The process of sampling and updating counters is done until we observe stability in the log likelihood [131]. At the termination the algorithm computes the posterior probability distribution, which reflects the BN parameters.

Algorithm 7 Gibbs sampling for modeling click counts

1: **Input** : DAG structured category hierarchy $G(V, E)$
2: observed categories for documents $\{C_1, ..., C_n\}$
3: importance scores for documents $\{\eta_1, ..., \eta_n\}$
4: **Output** : Parameters of BN
5: Create training instances by repeating documents:
6: $D = \{d_{1,1}, \ldots, d_{1,\eta_1}, d_{2,1}, \ldots, d_{2,\eta_2}, ...\}$
7: Let $X_{di} \in \{0, 1\}$ denote the current assignment of category random variable X_i for document $d \in D$
8: Let $X_{d\pi(i)} \in \{0, ..., 2^{|\pi(i)|} - 1\}$ be a variable representing the configuration of parents of X_i in document d.
9: ▷ Set observed categories and all ancestors to true:
10: Set $X_{di} = 1$ if $X_i \in C_d \vee X_i \in \pi^*(C_d)$
11: Initialize $X_{di} \sim \text{Uniform}(\{0, 1\})$ for all $d \in D$ and $i \in V$
12: Initialize counts:
$N_{iJ} = \sum_d 1(X_{di} = 1 \wedge X_{d\pi(i)} = J)$
$N_J = \sum_d 1(X_{d\pi(i)} = J)$

13: **for** $d \in D$ **do**

14: **for** $i \in V$ **do**

15: **if** $X_i \notin latentVariableSet(d)$ **then**

16: **continue**

17: **end if**

18: ▷ Remove current assignment to X_{di} from N_{iJ}

19: $J = X_{d\pi(i)}$

20: **if** $X_{di} = 1$ **then**

21: $N_{iJ} = N_{iJ} - 1$

22: **end if**

23: $N_J = N_J - 1$

▷ Re-sample X_{di}

24: $X_{di} \sim \text{Bernoulli}\,(P)$, where

$$P \propto \beta_{iJ} \prod_{k \in \text{children}(i)} \beta_{k\pi(k)}^{\mathbb{1}(X_{dk}=1)} \left(1 - \beta_{k\pi(k)}\right)^{\mathbb{1}(X_{dk}=0)}$$

and where $\beta_{iJ} = \frac{N_{iJ}+\alpha_0}{N_J+\alpha_1}$

25: Constrain X_{di} in the following cases:

- Set $X_{di} = 1$ if

  $\exists_{k \in \text{children}(i)}\ X_{dk} = 1 \wedge \forall_{l \in \pi(k)}\ X_{dl} = 0$

- Set $X_{di} = 0$ if

  $\neg\exists_{k \in \text{children}(i)}\ X_{dk} = 1$

▷ Add new assignment of X_{di} to N_{iJ}

26: **if** $X_{di} = 1$ **then**

27: $N_{iJ} = N_{iJ} + 1$

28: **end if**

29: $N_J = N_J + 1$

30: **end for**

31: **end for**

32: If not converged, goto 13

From the Gibbs Sampler's samples (or from the posterior distribution) we compute the

marginal probabilities of every category node in the $G(V;E)$ as

$$p(X_i) = \frac{\sum_J N_{iJ}}{\sum_{i,J} N_{iJ}} = \frac{\text{Number of documents with } X_i = 1}{\text{Total number of documents}} \tag{6.12}$$

This probability estimate reflects the importance of a category node and will be used to determine the appropriate sub DAG using an algorithm outlined in section 6.5. Before presetting the sub DAG selection algorithm, we present another approach based on EM to estimate the importance scores of the category nodes in the next Section.

6.4 EM based Parameter Estimation

Unlike the previous approach where we treated the DAG as a BN, in this approach we focus on the paths in the DAG from the root to the leaves. We assume that the probability of generating a document at the leaf node is proportional to the product of edge probabilities on the path. Given the category DAG with documents having importance scores at the leaf level, our goal is to estimate the edge probabilities. The leaf nodes' (documents') importance scores are observed. The probabilities of paths (categories) leading to the leaf nodes from the root are hidden. We want to estimate the probability of a path given the observations. In the Figure 6.2 we illustrate edge parameters which are the conditional probabilities. Given an edge going from node j to k, the probability of the edge is the probability of reaching node k given the node j. Similarly, the probability of reaching a leaf document node l given the category node u is probability of edge going from u to l. We segregate these two probabilities in order to model the category-category and document-category assignments. We estimate these probabilities (parameters of the model) by applying the EM algorithm. Before describing E and M steps, we list the notations that we use:

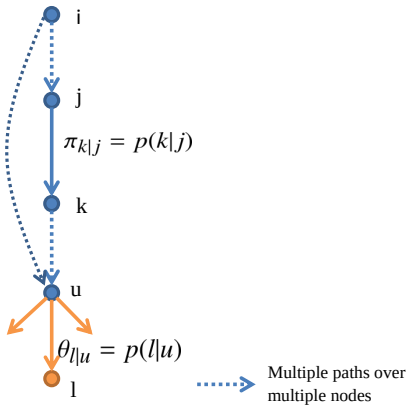


Figure 6.2: Illustration of path and edge parameters

$\mathcal{P}$	A path from the root to a category having only documents. Note, a path consists of only category nodes.
$v(i)$	i^{th} node on path $\mathcal{P}$
start($\mathcal{P}$)	Starting node of path $\mathcal{P}$
end($\mathcal{P}$)	Ending node of path $\mathcal{P}$
m($\mathcal{P}$)	Length of path $\mathcal{P}$
l	A leaf node, which is a document.
(j, k)	An edge between the categories j and k. The category j is the parent of category k.
$p(\mathcal{P}\|l)$	Probability of a path $\mathcal{P}$ given a leaf node l.
$\pi_{k\|j} = p(k\|j)$	Probability of edge (j, k). In other words, probability of reaching child k from the parent j.
$\Theta_{l\|k} = p(l\|k)$	Probability of generating document l from a given category k.
$\Theta_{l\|\mathcal{P}} = p(l\|\mathcal{P}) = p(l\|\text{end}(\mathcal{P}))$	Probability of generating document l for a given path $\mathcal{P}$. It is same as the probability of generating l from the last category 'end($\mathcal{P}$)' of path $\mathcal{P}$.
N_l	The importance score of the leaf (document) node l.

Table 6.1: Notations used in this section

6.4.1 E-Step:

In E step, we estimate $p(\mathcal{P}|l)$, the probability of path $\mathcal{P}$ to a given leaf node (document) l. Using Bayes rule, this can be written as:

$$p(\mathcal{P}|l) = \frac{p(l|\mathcal{P})\, p(\mathcal{P})}{\sum_{\mathcal{P}'} p(l|\mathcal{P}')\, p(\mathcal{P}')}$$

The probability of a path can be decomposed as the product of probabilities of edges on the path. Applying this to the above equation gives us the following quantity.

$$p(\mathcal{P}|l) = \frac{\Theta_{l|\mathcal{P}} \prod_{i=1}^{m(\mathcal{P})} \pi_{v(i)|v(i-1)}}{\sum_{\mathcal{P}'} \Theta_{l|\mathcal{P}'} \prod_{i=1}^{m(\mathcal{P}')} \pi_{v(i)|v(i-1)}} \tag{6.13}$$

While estimating the E step (that is, $p(\mathcal{P}|l)$), the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ are held at the values estimated from M step.

6.4.2 M-Step:

In M step, we estimate the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ by holding the path probabilities $p(\mathcal{P}|l)$ at the values estimated from E step. That is,

$$\pi_{k|j} = \frac{\sum_l N_l \sum_{\mathcal{P}:(j,k)\in\mathcal{P}} p(\mathcal{P}|l)}{\sum_l N_l \sum_{\mathcal{P}:j\in\mathcal{P}} p(\mathcal{P}|l)} \tag{6.14}$$

$$\Theta_{l|\mathcal{P}} = \frac{N_l \sum_{\mathcal{P}':\text{end}(\mathcal{P}')=\text{end}(\mathcal{P})} p(\mathcal{P}'|l)}{\sum_{l'} N_{l'} \sum_{\mathcal{P}':\text{end}(\mathcal{P}')=\text{end}(\mathcal{P})} p(\mathcal{P}'|l')} \tag{6.15}$$

The E and M steps are repeated until the convergence, at which the model parameters would have been estimated.

6.4.3 Downward-Upward Algorithm

Estimating model parameters $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ in EM algorithm (in equations 6.13,6.14,6.15) requires enumerating all the paths in the DAG. In a moderately big DAG, the number of paths can be prohibitively large to process on a computer. For for example, the category DAG in Wikipedia covering the documents in "Ambient" disambiguation page has approximately 20 billion paths. Allocating memory to store all the path probabilities and iterating through all the paths for computing various quantities on a moderate capacity computer (16 core, 32 GB RAM) proves to be impossible. This leads us to discover a'Downward-Upward' algorithm (inspired by Inside-Outside algorithm [8]) to efficiently compute the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$, without enumerating all the paths. At each EM step, the "Downward-Upward" algorithm first traverses the DAG from root to leaves and then from leaves to the root, by propagating aggregated quantities downwards and then upwards.

By substituting Equations 6.14 and 6.15 into Equation 6.13 we get

$$p^{(t)}(k|j) = \frac{\sum_l N_l \frac{1}{Z_l} \sum_{\mathcal{P}:(j,k)\in\mathcal{P}} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right)}{\sum_l N_l \frac{1}{Z_l} \sum_{\mathcal{P}:j\in\mathcal{P}} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right)} \tag{6.16}$$

where $z_l = \sum_{\mathcal{P}'} p^{(t-1)}(l|\text{end}(\mathcal{P}')) \prod_{(r,s)\in\mathcal{P}'} p^{(t-1)}(s|r)$ and t is the iteration index.

We define two functions $\alpha(j)$ and $\beta(j)$ as follows:

- $\alpha(j)$ is the total probability of reaching node j from the root node through all the paths from root to j. It can be recursively defined as

$$\alpha(j) = \begin{cases} \sum_{a\in\text{parents}(j)} p(j|a)\,\alpha(a) & \text{if } j \text{ is not the root node} \\ 1 & \text{if } j \text{ is the root node} \end{cases} \tag{6.17}$$

  This can be calculated efficiently by making one pass from the root to the leaves, accumulating the values at each node.

- $\beta(j)$ is the fraction of value flowing to node j from the leaf node l through all the paths between j and l. It can be recursively defined as

$$\beta(j) = \begin{cases} \sum_{b \in \text{children}(j)} p(b|j)\,\beta(j) & \text{if } j \text{ is not a leaf node} \\ \frac{N_j}{\alpha(j)} & \text{if } j \text{ is a leaf node} \end{cases} \tag{6.18}$$

  This can be calculated efficiently by making one pass from the leaves to the root, accumulating the values at each node.

The equation 6.16 can be rewritten in terms of α and β as follows:

$$\begin{aligned} p^{(t)}(k|j) =& \frac{\alpha(j)\,p^{(t-1)}(k|j)\,\beta(k)}{\alpha(j)\,\beta(j)} \\ =& p^{(t-1)}\frac{\beta(k)}{\beta(j)} \end{aligned} \tag{6.19}$$

The proof is given in Appendix B.

Note that, $\pi_{k|j} = \lim_{t\to\infty} p^{(t)}(k|j)$ and $\Theta_{l|\mathcal{P}} = \lim_{t\to\infty} p^{(t)}(l|\text{end}(\mathcal{P}))$

Algorithm8 outlines the Downward-Upward algorithm.

Algorithm 8 Downward-Upward Algorithm

1: Initialize edge probabilities uniformly
2: **while** not converged **do**
3: Downward Propagation: Compute α for each leaf node as in Equation 6.17
4: Upward Propagation: Compute β for each node as in Equation 6.18
5: Update edge probabilities as in Equation 6.19
6: **end while**

From the estimated edge probabilities we further compute the marginal probabilities of every category node X_i in the $G(V;E)$ as

$$p(X_i) = \alpha(i) \tag{6.20}$$

This probability score reflects the importance of a category node. Combining this score with the entropy of the children, we rank and chose top K category nodes as explained in next section.

6.5 Constructing Sub-DAGs

Once the marginal probabilities $\{p(X_i)\}_{i=1}^{|V|}$ of category nodes $\{X_i\}_{i=1}^{|V|}$ of the DAG $G(V;E)$ have been estimated using the Gibbs Sampling (Equation 6.12) or EM based (Equation 6.20) algorithms, we determine the importance of a category node X_i through its marginal probability and how important its children are. If a node has high marginal probability and all its children too have high marginal probabilities, then we score such a node highly. Naturally, entropy of children s' marginal probabilities is an indication of how informative are the children of a node. Formally we define the entropy over a node X_i's children as follows:

$$H(X_i) = \sum_{k \in \text{children}(X_i)} \bar{p}(X_k) \log(\bar{p}(X_k))$$

$\bar{p}(X_k)$ is the normalized marginal probabilities of children (X_i), that is, $\bar{p}(X_k) = \frac{p(X_k)}{\sum_{j \in \text{children}(X_i)} p(X_j)}$

The rank of a node X_i is defined as follows:

$$r(X_i) = p(X_i) \times H(X_i)$$

Given a budget K, we choose top K ranked nodes and create edges between the nodes X_i and X_j if X_i is ancestor of X_j in DAG $G(V;E)$. This produces a sub-DAG of K nodes that is compact and representative of the of the document collection. In the experimentation section we give a heuristic to estimate the value of K from the training set.

6.6 Experimental Results

To validate our approach, we make use of the Wikipedia category structure as a category DAG and apply our technique to the task of automatic generation of Wikipedia disambiguation pages. We pre-processed the category graph to eliminate the cycles in order to make it a DAG. In Appendix A we outline this procedure. Each Wikipedia disambiguation page is manually created by Wikipedia editors by grouping a collection of Wikipedia articles into several groups. Each group is then assigned a name, which serves as a category for the group. Typically, a disambiguation page divides around 20-30 articles into 5-6 groups. Figure 6.3 shows the disambiguation page for "Apple". Our goal is to measure how accurately we can recreate the groups for a disambiguation page and label them, given only the collection of articles mentioned in that disambiguation page (when actual groupings and labels are hidden.)

6.6.1 Datasets

We parsed the contents of Wikipedia disambiguation pages and extracted disambiguation page names, article groups and group names. For each article, we extracted click count information from Wikipedia's click count logs. We collected approximately 800 disambiguation pages that had at least four groups on them. Wikipedia category structure is used as the category DAG. We eliminated few administrative categories such as "Hidden Categories" , "Articles needing cleanup" , and the like. The final DAG had approximately 1M categories and 3M links.

Using disambiguation page title as a keyword query, we retrieved all the Wikipedia articles having those keywords in their title. For each of these articles, we also extracted click counts from the click-logs published by Wikimedia. We eliminated about 30% of the articles having low click counts. Remaining articles are then added to the disambiguation page. Note that, for these added articles, we do not know the actual group in the disambiguation page they belong to. Hence, we do not use these articles while computing the metrics (described in next section.) We only use the articles that are grouped under a disambiguation page by the human editors. However, adding queried articles are important because: (i) more data makes our Gibbs sampling and EM based algorithms

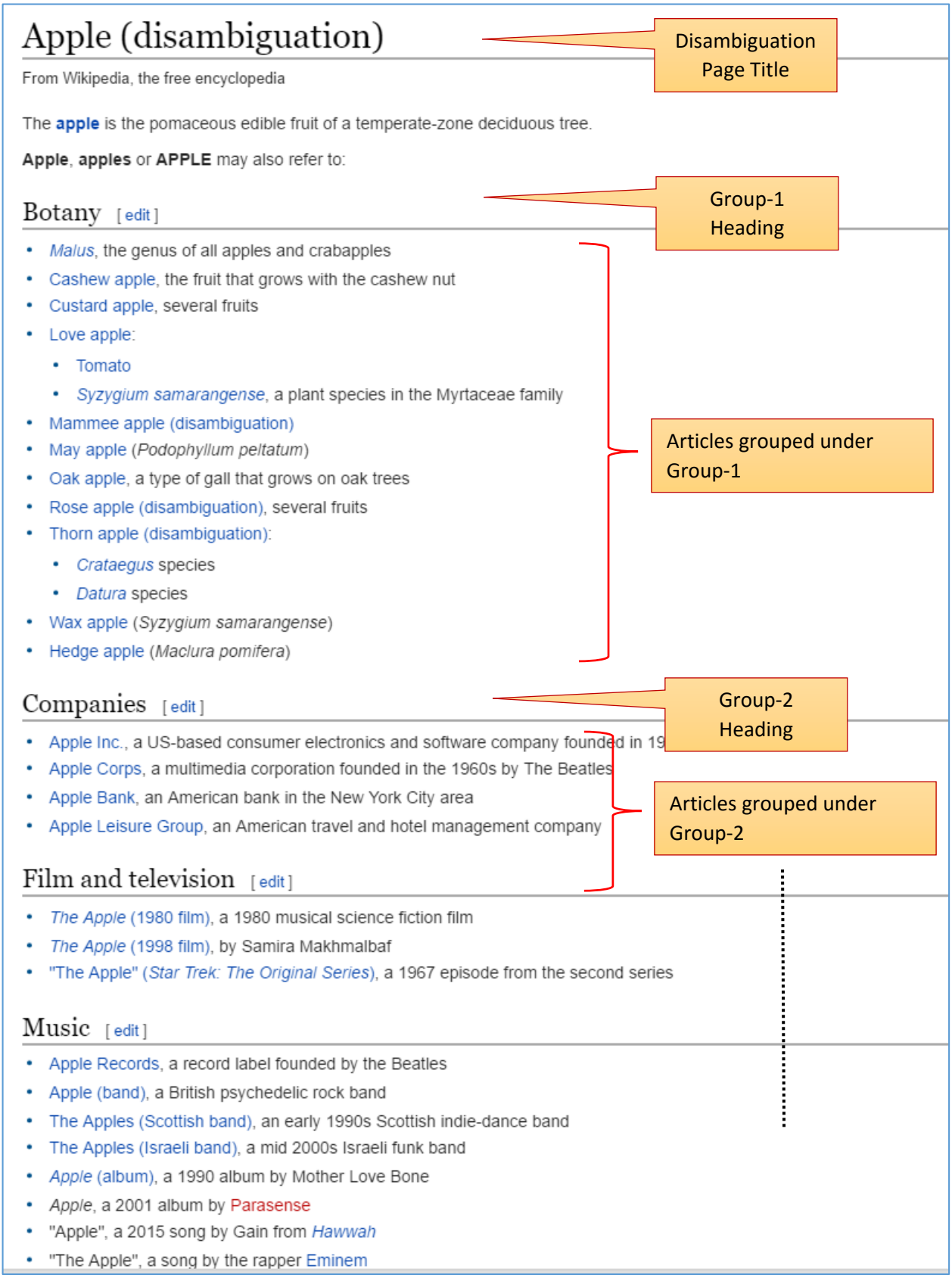


Figure 6.3: Wikipedia disambiguation page for Apple.

produce better results, and (ii) in practice (during inference time) we are only given a disambiguation keyword and our task is to generate the disambiguation page for it. We then have to query the Wikipedia articles using the the disambiguation keyword.

6.6.2 Evaluation Metrics

While the Wikipedia disambiguation page data set provide a large collection of human labeled data, it poses two challenges for the evaluation of our methods; (i) Every group of articles on the Wikipedia disambiguation page is assigned a name by the editors. Unfortunately, these names may not correspond exactly to the Wikipedia category names. For example, one of the groups on the "Matrix" disambiguation page has a name "Business and government" and there is no Wikipedia category by that name. However, the group names generated by the automated method are these of the Wikipedia categories (which forms our category DAG). In addition, there can be multiple relevant names for a group. For example, a group on a disambiguation page may be la belled "Calculus" , but an algorithm may correctly assign it to "Vector Calculus". (ii) While the Wikipedia disambiguation page group names (in most of the cases) form a single level hierarchy, our methods create a DAG structured hierarchical group names. Hence we cannot evaluate the accuracy of the automated methods just by matching the generated group names to those on the disambiguation page. To alleviate this problem, we adopt cluster-based evaluation metrics. We treat every category node of the sub DAG generated by our algorithm as a cluster of articles. All articles in this cluster are reachable from a path originating form the category node. These are considered as inferred clusters for a disambiguation page. We compare them against the actual grouping of articles on the Wikipedia disambiguation page by treating those groups as true clusters. We can now adopt hierarchical cluster evaluation metrics – F- Score measure and Entropy – from the works of Zhao et al. [161]. For each disambiguation page in the dataset, we compute every metric score and then average it over all the disambiguation pages.

6.6.2.1 F-Score metric

F-Score measure identifies for every class of documents, a node in the hierarchical DAG that best matches it. Note that in our setup, the class of a document is the group name under which the document is listed on the Wikipedia disambiguation page. The match of a cluster to the class is measured using the F_1 value that combines the standard precision and recall functions used in information retrieval. Specifically, given a particular class L_r of size n_r and a particular cluster S_i of size n_i , suppose n_i^r documents in the cluster S_i belong to L_r, then the F_1 value of this class and cluster is defined to be

$$F_1(L_r, S_i) = 2 * R(L_r, S_i) * P(L_r, S_i) / \left(R(L_r, S_i) + P(L_r, S_i)\right)$$

where $R(L_r, S_i) = n_{ir}/n_r$ is the recall value and $P(L_r, S_i) = n_{ir}/n_i$ is the precision value defined for the class L_r and the cluster S_i. The F-Score of class L_r is the maximum F value seen at any node in the DAG G. That is,

$$\text{FScore}(L_r) = \max_{S_i \in G} \; F(L_r, S_i) \tag{6.21}$$

The F-Score of the entire DAG is computed as the sum of the individual class specific F-Scores weighted according to the class size. That is,

$$\text{FScore} = \sum_{r=1}^{c} \frac{n_r}{n} \text{FScore}(L_r)$$

where c is the total number of classes. If the clustering solution is perfect, then every class has a corresponding cluster containing the same set of documents in the resulting hierarchical DAG. This results in perfect F-Score of one. In general, the higher the F-Score values, the better the clustering solution is.

Note that, this metric does not consider the DAG size or structure while computing F-Score. Hence it is trivially possible to maximize F-Score in our algorithm just by outputting entire DAG-structured hierarchy, instead of finding optimal sub-DAG. To overcome this, we also evaluate on another metric -an Entropy based measure.

6.6.2.2 Entropy metric

In the F-Score measure, evaluation of the overall quality of a hierarchical DAG takes place from a small subset of its nodes. To overcome this problem, Entropy measure is defined, which takes into account the distribution of the documents in all the nodes of the tree. Given a particular node S_r of size n_r, the entropy of this node is defined to be

$$E(S_r) = -\frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r}$$

where q is the number of classes in the dataset and n_r^i is the number of documents of the i^{th} class that were assigned to the r^{th} node. Then, the entropy of the entire DAG G is defined to be

$$E(G) = \sum_{r=1}^{p} \frac{1}{p} E(S_r) \tag{6.22}$$

where p is the number of non-leaf nodes of the DAG G. In general, the lower the entropy values the better the clustering solution is.

6.6.3 Methods Compared

Closest to our technique is a score propagation technique [56, 156] where each node passes scores to its parent nodes by equally dividing the message among its parents. Equal division is one of the schemes where every node thinks all its parents are equally likely (akin to a uniform prior), which is acceptable in the absence of any other information to differently weight the parents. Note that in our work we do not look into the text of documents/category nodes, hence we do not have any other information on the likelihood of a parent. The scores originate from the leaf (document) nodes and propagate upwards towards the root. The leaf nodes are initialized with the score values equal to their importance counts. After the score propagation passing algorithm stabilizes, the marginal probability of a node is proportional to the total amount of score passed through that node. Since the scores originate from the document nodes and are initialized to the importance scores of the documents, the marginal probabilities of the nodes reflect the importance scores of the documents. We then apply our ranking method to rank and build a sub-graph as explained in Section 6.5. We call this technique as "Equal-Weighting."

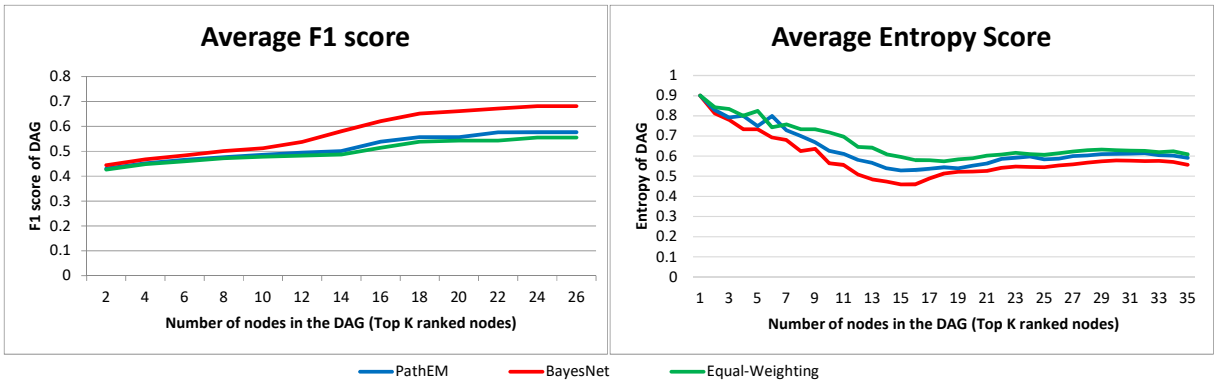


Figure 6.4: Average F1 and Entropy scores vs DAG size (higher the F1 better; lower the Entropy better)

We call our method that we described in Section 6.3, which is based on treating the DAG structured hierarchy as a Bayesian Network and estimating its parameters via Gibbs sampling as "Bayes Net". And, the method described is Section 6.4, which is based on sampling paths and estimating parameters based on EM as "PathEM"

6.6.4 Evaluation Results

Our first evaluation is based on the F1 metric. Figure 6.4 first half shows the F1 metric for various values of top K ranked nodes, that is, sub-DAG of size K nodes. This is the average F1 score from 800 disambiguation pages. In Figure 6.5 we show the F1 scores for the DAGs of 4 disambiguation pages. By construction, the F1 score monotonically increases with the size of the DAG. Hence, the more nodes we add to the DAG, the better is the F1 score. Therefore, we should not compare the maximum F1 scores of different algorithms, which obviously well all be equal (a DAG of maximum size.) The F1 scores initially increase rapidly due to the clustering formation that takes place with the addition of each node to the DAG, improving the F1 score. After a certain size (around $K = 15$) the addition of new nodes does not change the clusters and F1 measures that rapidly due to the max F1 in Equation 6.21 has already found a good node that clusters the documents close to the true clusters. Hence F1 scores start plateauing. Therefore, we recommend to compare the techniques for the range $K = 15$ to 20.

The Bayes-net method performs better than PathEM, which in turn performs better than Message-passing. Since the Gibbs sampling employed in Bayes-net is able to come out of

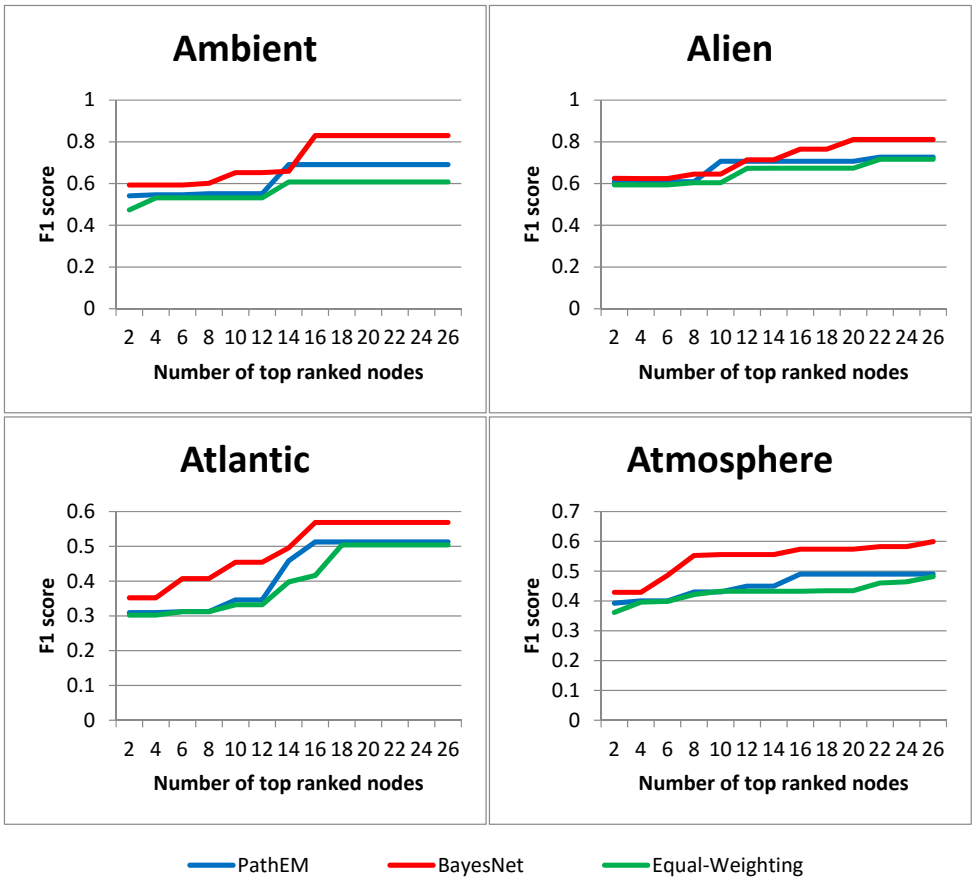


Figure 6.5: F1 scores of DAGs of 4 Disambiguation pages at various DAG sizes (higher score is better)

local minimum due to the sampling nature, it performs better than PathEM, which often gets stuck into local minimum. Message-passing performs poorly due to the assumption it makes in dividing the messages equally among the parents. This assumption does not seem to perform well because, often a particular parent is more important than other parents. For for example, the category "Sports" is more relevant parent to the category "Soccer" than the parent category "21st Century Players."

Next we evaluate techniques using the Entropy based measure. The right part of the Figure 6.4 shows the Entropy metric for various values of top Knodes, that is, sub-DAG of size K nodes. This is the average Entropy score from 800 disambiguation pages. In Figure 6.6 we show the Entropy scores for the DAGs of 4 disambiguation pages. Entropy score initially decreases (smaller the Entropy score, better) up to K around 15 and then starts to increase. Due to the clustering formation improves with the addition of each node initially, the Entropy score starts to decline. However, when more nodes are added

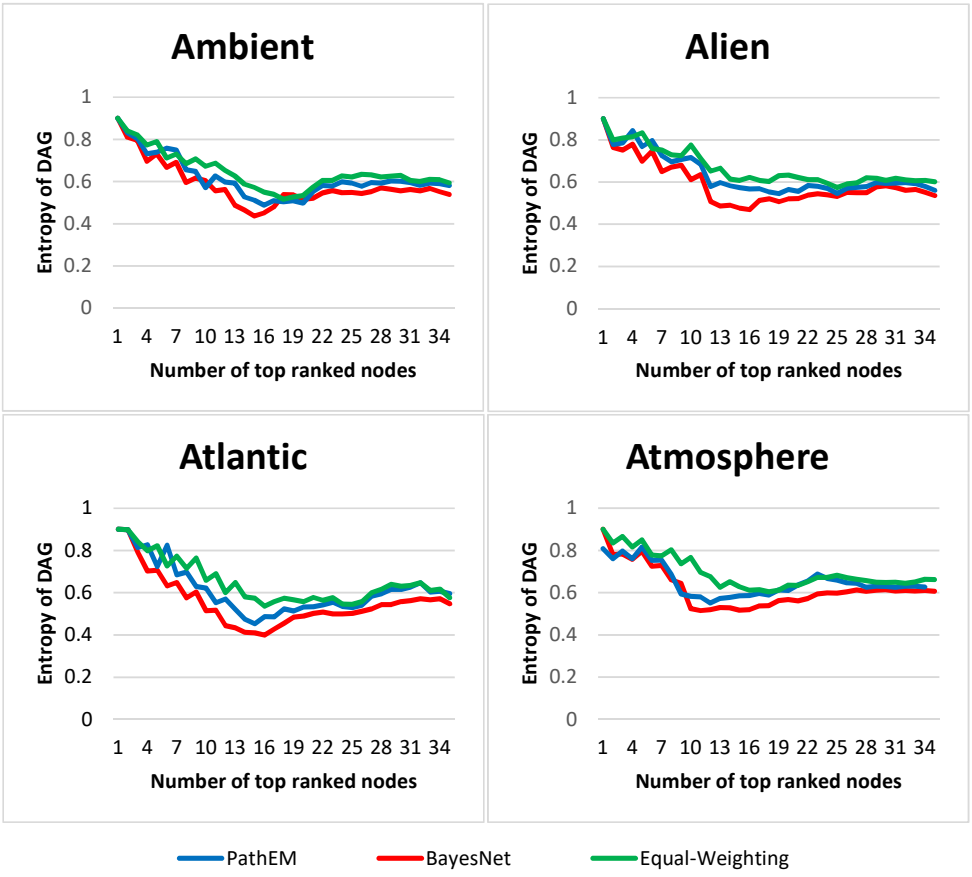


Figure 6.6: Entropy scores of DAGs of 4 Disambiguation pages at various DAG sizes (lower score is better)

(that is, as K increases beyond 20) the later nodes do not induce as good clusters as initial nodes, making the overall Entropy score in Equation 6.22 to increase. However, with large K (greater than 40 or 50) we observe decrease in Entropy once again due to the the addition of "fine-grained" (close to the leaf) categories, which are often associated with single document, resulting in very low Entropy, thus reducing the overall Entropy in Equation 6.22. Therefore, we recommend to compare different techniques around the first mini ma that happens around $K = 15$. At this K value, we see that Bayes-net performs better than PathEM, which performs better than Message-passing, the reason being the same as explained earlier in F1 measure case.

Figure 6.7 shows an output (sub-DAG) of Bayes-net algorithm run on the "Ambient" disambiguation page.

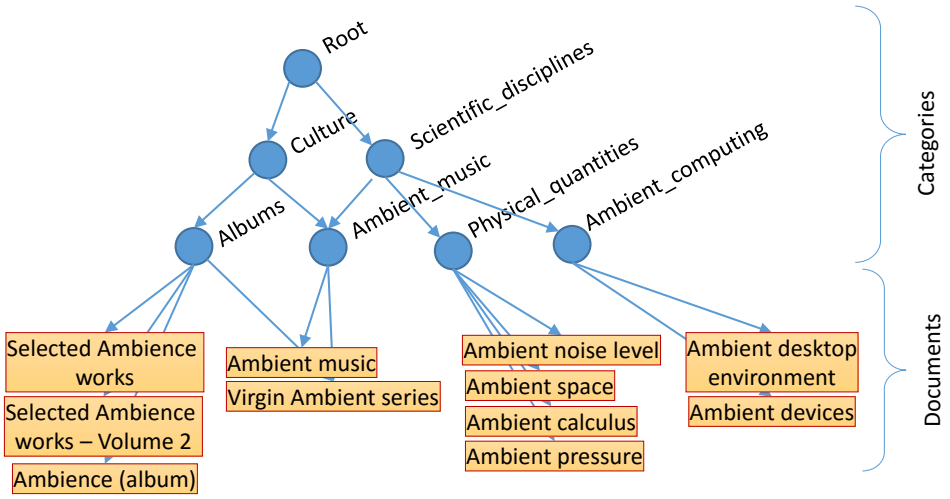


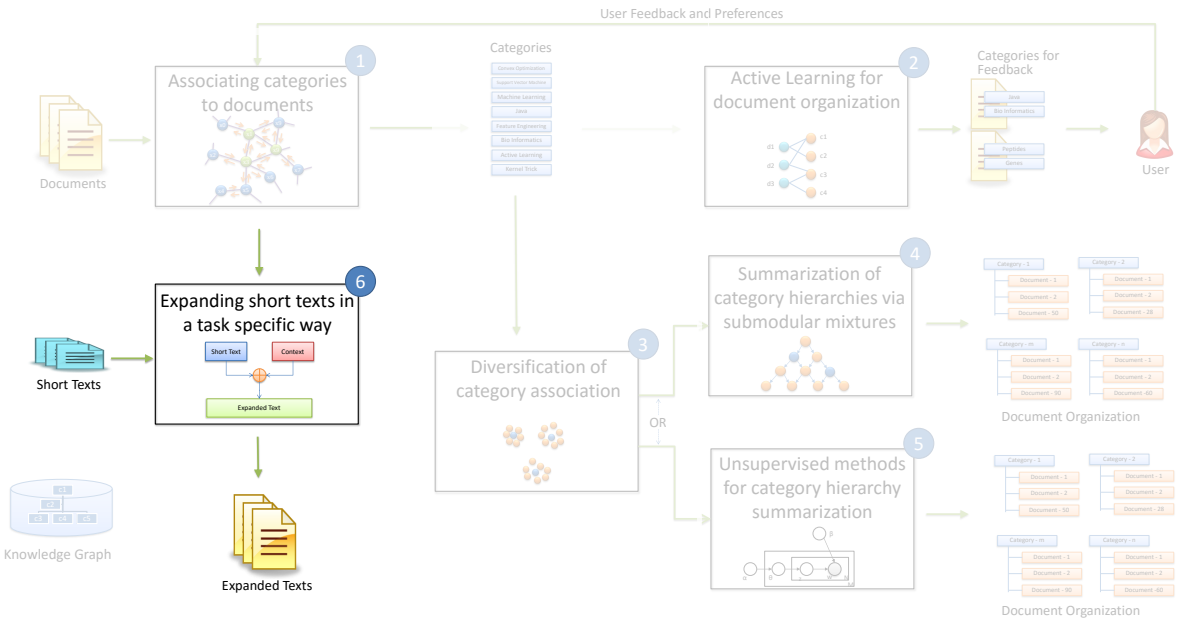
Figure 6.7: Sub-DAG (of Wikipedia categories) generated for "Ambient" disambiguation page by BayesNet algorithm.

6.7 Conclusion

We investigated a problem of generating a sub-DAG of categories over a massive DAG-structured category hierarchy such that the sub-DAG produced represents the importance scores of the documents. This representation is characterized through a generative model that learns its parameters such that, repeated sampling of a path ending in a document from the sub-DAG is able to generate the distribution of observed importance scores. Like in the previous chapter, through this work we are able to reduce a large number of categories that get accumulated from all the documents in a document collection, thus addressing the "over-specified" categorization problem.

Chapter 7

Organization of Short Texts via Expansion*



*__Published__: "A Framework for Task-specific Short Document Expansion", Ramakrishna Bairi, Raghavendra Udupa, Ganesh Ramakrishnan, In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, Indianapolis, USA, Oct 24-28, 2016.

Chapter Summary

Collections that contain a large number of short texts are becoming increasingly common (for example, tweets, reviews, and the like.) As these collections grow in number and size, effectively organizing them becomes an important activity. Document organization and other analytical tasks (such as classification, clustering, and the like.) involving short texts could be challenging due to the lack of context and owing to their sparseness. An often encountered problem is low accuracy on the task[1]. A standard technique used in the handling of short texts is expanding them before subjecting them to the task. However, existing works on short text expansion suffer from certain limitations: (i) they depend on domain knowledge to expand the text; (ii) they employ task-specific heuristics; and (iii) the expansion procedure is tightly coupled to the task. This makes it hard to adopt a procedure, designed for one task, into another. We present an expansion technique – TIDE – that can be applied on several Machine Learning, NLP and Information Retrieval tasks on short texts (such as short text classification, clustering, entity disambiguation, and the like) without using task specific heuristics and domain-specific knowledge for expansion. At the same time, our technique is capable of learning to expand short texts in a task-specific way. That is, the same technique that is applied to expand a short text in two different tasks is able to learn to produce different expansions depending upon what expansion benefits the task's performance. To speed up the learning process, we also introduce a new technique called *block learning*. Our experiments with classification and clustering tasks show that our framework improves upon several baselines according to the standard evaluation metrics which includes the accuracy and normalized mutual information (NMI). .

[1]Hereafter, we use the term *task* to refer to the short text organization. However, the task can very well mean other ML/IR tasks such as classification, clustering, etc. Our technique presented in this chapter is very generic and can be applied to any of these tasks

7.1 Introduction

With the rapid development of the Internet, Web users are generating an increasing number of short texts, including tweets, search snippets, product reviews and the like. Successfully processing them becomes increasingly important to many IR and Machine Learning applications. However, short texts are quite different from the traditional documents due to their being short and sparse. Hence, conventional machine learning and IR algorithms cannot apply to short texts directly. Owing to lack of context in the short text, it becomes challenging to build a representative feature vector to use it in a machine learning task such as document organization, classification, clustering, ranking or searching, to name a few.

Note that, although this thesis is a study on the organization of digital documents (which include short texts too,) in this Chapter we treat it as an IR/ML task. The techniques presented in this Chapter are applicable to almost any IR/ML task, including document organization. Hence we do not make a distinction here. Throughout this Chapter, we use the term *task* to refer the document organization task and any other ML/IR task.

Existing approaches [109, 34, 20, 121, 157] attempt to address challenges associated with analytic tasks on short texts by introducing more context through expansion techniques. Broadly, these approaches follow one of two techniques. The first one is to employ a search engine (for example, Google) or an IR system (for example, Lucene[2], Solr[3]) to use the short text as a search query and expand the short text using the top search results [20, 121, 157]. The other technique is to build a topic space and to project each short text into that space. These topics enrich the context of the short text [122, 9, 109, 34].

Although querying search engines using short texts can produce good expansions, it may not be an ideal solution for many applications, given its online nature and given restricted programmatic access to search engines. Though querying a local IR system is fast, we learn from experiments that the top result that is returned by the search engine or the IR system may not always be the best for the task.

[2] `https://lucene.apache.org/`

[3] `http://lucene.apache.org/solr/`

Using a predefined topic space may not be a feasible solution because there may not be predefined topics/taxonomy for certain applications, domains and languages. A solution that is based on latent topic space is preferable because the latent topics can be generated from a large corpus that is available in the domain of an application. However, solutions that are based on topic space are restricted to certain types of applications. It is difficult to generalize these solutions in different types of applications. For example, though tasks such as classification of short texts can be efficiently solved by these approaches [109, 34], the approaches may not be suitable for tasks on learning to rank short texts. Primarily, this is due to the bag-of-words model assumption that these approaches make while expanding the short texts, that is, the pseudo topic names that are appended to each short text as additional words help the classifiers to learn discriminative weights for each class, thus boosting the classification accuracy. However, if the task is the retrieval and ranking of short texts, the solution of adding pseudo topic names may not help. One of the main characteristics of our approach is that we do not make such assumptions, thus our approach generalizes well to other tasks.

A large body of research has been directed at using sources of structured semantic knowledge such as Wikipedia, DBPedia and WordNet for document expansion [9, 122, 66, 52, 54, 90]. On the other hand, distributional semantic approaches are based on the intuition that words appearing in similar contexts tend to have similar meanings. One such approach is word vectors—also referred to as word embeddings— which has recently seen a surge of interest since new ways of computing them efficiently [99, 106] have become available. Word embeddings provide a way to expand documents by attaching average or augmented word vectors of the terms in the document [154]. While all these approaches provide ways to expand short texts, the expansion itself is independent of the task that uses them; there is no guarantee that such an expansion would benefit the task.

To address these shortcomings we develop an approach that has the following characteristics: (i) It makes fewest assumptions about the task; the task should be able to consume the suggested expansions and produce a measurable performance metric (such as task accuracy), nothing else. (ii) It uses an expansion technique that generalizes well across different tasks. (iii) It expands the short texts selectively, that is, a short text is expanded only if such an expansion helps the task; otherwise, it is not expanded. Blindly expanding

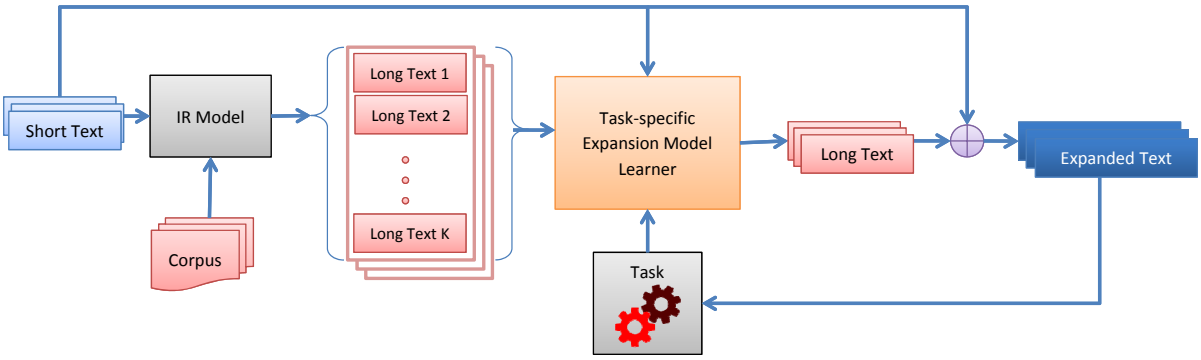


Figure 7.1: Architecture of expansion model

all the short texts – as done in earlier approaches – in fact degrades the task's performance. (iv) It is able to learn a model that produces the expansions specific to the task. However, the features used by the model generalizes well to all the tasks, thus making the approach applicable to many tasks. In section 7.2.3 we present different classes of feature functions which can be used across multiple tasks. Figure 7.1 depicts overall architecture of our approach/framework. As in many earlier approaches [109, 34] we use a corpus that contains relevant *long text* articles. In section 7.3.3, we comment more on choosing the right corpus. Using short text as a query, a language model or an IR system initially selects K long texts from the corpus as expansion candidates. Our observation shows that we can usually find the best long text for expansion within the top $10 - 20$ results. We then learn an expansion model that is specific to the task (by taking task's performance metric as an input) for selecting one of the K candidate long texts as the expansion of the short text. The goal of learning is to produce a mapping of the short texts to the long texts which best improves the task accuracy. Note that the framework only requires an accuracy value from the task and does not exploit any other characteristics of the task or domain. Hence the framework is applicable to a wide variety of tasks.

In particular, our framework learns a best mapping function $\mathcal{M} : \{t_{\text{short}}\} \times \mathcal{T} \rightarrow \{t_{\text{long}}\}$ to maximize the task $\mathcal{T}$ accuracy. We can present this function in an alternate form as

$$\mathcal{M}\left(t_{\text{short}}\right) = \underset{t_{\text{long}} \in \mathcal{C}}{\text{argmax}} \ \text{Sim}\left(t_{\text{short}}, t_{\text{long}}; \mathcal{T}\right)$$

which is a mapping function that finds a maximally similar long text t_{long} to a short text t_{short}, given the task $\mathcal{T}$. The similarity $\text{Sim}(\cdot)$ is captured through a variety of functions that are designed from the IR, topic model and embedding techniques and generalize well

to many tasks. The function $\mathcal{M}$ needs to be learned specific to the task. There are no additional training data for learning $\mathcal{M}$. We have designed a novel learning technique to learn $\mathcal{M}$ jointly with the task during the training phase of the task, thus making $\mathcal{M}$ task specific. In section 7.2.4, we elaborate our learning technique using a derivative-free optimization technique that is known as BOBYQA (Bound Optimization BY Quadratic Approximation) [110] along with our proposed *block learning* approach.

Our contributions can be summarized as follows:

- A framework for task-specific document expansion that can be adopted by many machine learning tasks that deal with the short texts
- The introduction of various classes of task agnostic feature functions (IR, topic model and embedding based features)
- A technique for learning a model over task agnostic features, using the task's data through the task itself, thus learning to expand a short text in a task-specific way.
- A *Block learning* technique for learning feature weights in blocks

7.2 Learning Task-specific Document Expansion

7.2.1 Problem Formulation

Let $\mathcal{D}_s = \{s_i\}_{i=1}^{n}$ be a set of n short texts, where s_i is the representation of i^{th} short text. For example, s_i can be a TF-IDF vector representation of the short text, or it can simply be a bag of words. The exact form of s_i is task-specific.

Let $\mathcal{T}(\mathcal{D}_s)$ be a task that operates on the short texts $\mathcal{D}_s$. For example, $\mathcal{T}$ can be a classification task and $\mathcal{D}_s$ is a set of short texts for training and testing the classifier. The underlying assumption is that task $\mathcal{T}$ can be run on the data $\mathcal{D}_s$ and produce a measurable result that indicates the goodness of the task, such as the accuracy of the task. In the classification task example that is mentioned above, we can measure the accuracy of the classifier that is trained on the training set and validated on the test set.

Other examples of the tasks include Clustering of short texts, Named Entity Recognition in short texts and Categorization of short texts using Wikipedia, to name a few.

Let $\mathcal{E}(\mathcal{T}(\mathcal{D}_s))$ be the empirical error in performing task $\mathcal{T}$. For example, in the classification task, $\mathcal{E}$ could be a "misclassification rate"; in the clustering task, it could be "degree of clustering disagreement". Furthermore, we assume that the error can be scaled to the [0,1] interval so that we can simply compute the accuracy of the task $\mathcal{T}$ as $1 - \mathcal{E}(\cdot)$.

A standard technique to improve the accuracy of the task $\mathcal{T}$ is to expand the short texts $\mathcal{D}_s$ into *bigger* texts. Let $\mathcal{D}_e = \{e_i\}_{i=1}^n$ be the expanded short texts, where e_i is a representation for the expanded i^{th} short text. For example., e_i can be a TF-IDF vector that represent the expanded text.

Our aim is to present a technique that expands the short texts $\mathcal{D}_s$ into long texts $\mathcal{D}_e$ such that, performing the task $\mathcal{T}$ on these expanded texts best improves the task accuracy (this equivalently reduces the task error rate.) Formally, by applying our technique, it is possible to achieve the following with a maximum difference between RHS and LHS.

$$\mathcal{E}(\mathcal{T}(\mathcal{D}_e)) \leq \mathcal{E}(\mathcal{T}(\mathcal{D}_s)) \tag{7.1}$$

To perform the expansion, we assume the existence of a right universal corpus $\mathcal{C}$ of articles. Such a corpus has to contain a large amount of articles in the domain of short texts, which covers the topics and concepts present in the short texts. These articles provide contexts to the short texts and help in expanding them. We refer to these articles as *long* texts. Our goal is to identify the best possible *long* text $l_i \in \mathcal{C}$ for every short text s_i. This long text l_i is then used to expand the short text s_i. We use the operator $\oplus$ to represent the expansion process of obtaining an expanded text representation of a short text from a long text. That is, $e_i = s_i \oplus l_i$. The exact process of this expansion depends on the task. For example, $\oplus$ may be a union of s_i and e_i, or it may be a weighted average of TF vectors. Mapping short texts $\{s_i\}_{i=1}^n$ to long texts $\{l_i\}_{i=1}^n$ is done in such a way that the task using the expanded short texts will have maximum improvement in accuracy. This makes the short text mapping/expansion task-specific. That means that the same short text may map to different long texts in different tasks, depending upon which mapping

makes the task better. However, our framework for expansion remains the same making it task agnostic.

In particular, we present a technique in this chapter to discover a mapping of $l_i = \mathcal{M}(s_i)$ to a long text $l_i \in \mathcal{C}$ for every short text $s_i \in \mathcal{D}_s$, such that Equation 7.1 is satisfied with maximum margin. Formally, we solve the following optimization problem.

$$\begin{aligned} \mathcal{M} = & \underset{\mathcal{M}}{\text{argmax}}\ \gamma \\ & \text{s.t. } \mathcal{E}(\mathcal{T}(\mathcal{D}_s \oplus \mathcal{M}(\mathcal{D}_s))) = \mathcal{E}(\mathcal{T}(\mathcal{D}_s)) + \gamma \\ & \gamma \geq 0 \end{aligned} \tag{7.2}$$

where $\mathcal{D}_e = \{e_i\}_{i=1}^n = \mathcal{D}_s \oplus \mathcal{M}(\mathcal{D}_s) = \{s_i \oplus \mathcal{M}(s_i)\}_{i=1}^n$

An optimum mapping $\mathcal{M}$ increases the difference between $\mathcal{E}(\mathcal{T}(\mathcal{D}_s \oplus \mathcal{M}(\mathcal{D}_s)))$ and $\mathcal{E}(\mathcal{T}(\mathcal{D}_s))$, thus maximizing the margin γ. Solving this optimization problem turns out to be difficult. In the following Sections we present a technique for approximately the solution to the optimization problem above. Our experiments on benchmark datasets show that our approach is very effective in practice and can produce results that are close to optimal.

7.2.2 Learning framework

Finding optimal mapping to solve Equation 7.2 is a combinatorial search problem. There are $|\mathcal{C}|^{|\mathcal{D}_s|}$ possible mappings of short texts $s_i \in \mathcal{D}_s$ to long texts $l_i \in \mathcal{C}$. However, not all of these mappings are meaningful. For example, mapping a short text on 'soccer" to an article on "photosynthesis" makes very little sense. To be a semantically correct expansion, the short text and its mapped long text have to be topically similar. Many IR techniques [92] have been successful in retrieving and ranking documents for short queries. They adopt language modeling techniques [79] to map short text queries to long text articles. This makes the language modeling approach a possible approximation to solve Equation 7.2.

In the language modeling approach to information retrieval, a multinational model $p\left(w|l_i\right)$ over terms is estimated for each document l_i in the collection $\mathcal{C}$ to be indexed and searched. This model is used to assign a likelihood to a short text $s_i = (w_1, w_2, ..., w_m)$. In the simplest case, each short text term is assumed to be independent of the other short text terms, so that the short text likelihood is given by $p\left(s_i|l_j\right) = \prod_{k=1}^{m} p\left(w_k|l_j\right)$. After the specification of a document prior $p(l_j)$, the a posterior i probability of a document is given by: $p\left(l_j|s_i\right) \propto p\left(s_i|l_j\right) p\left(l_j\right)$, and is used to rank the documents in collection $\mathcal{C}$.

The language model approach has the following limitations in mapping a short text s_i to a long text l_i: (i) it always produces the same mapping irrespective of the task; (ii) it does not take the task accuracy into account while mapping; and (iii) our observations show that the top ranked result of a language model need not always be the best mapping.

In order to overcome these limitations, we propose a two-step process for mapping. First, we use a language model to retrieve the top K candidate for the mapping of long texts $\left\{l_i^k \in \mathcal{C}\right\}_{k=1}^{K}$ for every short text s_i. Second, we define a mapping $\mathcal{M}_w$ to map a short text s_i to the best long text l_i from the K candidate long texts, such that using l_i to expand s_i best improves the accuracy of the task. $\mathcal{M}_w$ is computed by solving the following linear model:

$$l_i = \mathcal{M}_w\left(s_i\right) = \underset{l \in \left\{l_i^k \in \mathcal{C}\right\}_{k=1}^{K}}{\operatorname{argmax}} \sum_f w_f \phi_f\left(s_i, l\right)$$

where, ϕ_f is a feature function that measures the similarity between a short and a long text and w_f is the weight of that feature function in the mixture of features. A bunch of feature functions designed from the proven methods of IR, topic model and embedding techniques are explained in Section 7.2.3. These feature functions are then combined by learning weights $W = [w_f]_{f=1}^{f=(\#\text{ of features})}$ for them in a task-specific manner, which is explained in Section 7.2.4.

7.2.3 Classes of Feature Functions for Mapping

In the below sections, we define a variety of feature functions that measure the similarity of a short text to a long text. We group them into three classes: (i) IR based, (ii) Topic Model based, and (iii) Embedding based.

In the following sections we use the notations s and l to represent short and long texts, respectively, and $|l|$ to denote the length of l. We drop the subscript i for brevity.

7.2.3.1 IR Based Features

IR-based features compute the relevance of a short to a long text using standard document similarity functions such as BM25, Cosine, and Bigram. These functions have been successfully adopted by the information retrieval community in order to retrieve documents using short queries [92]. Accordingly, we define features such as $\phi_{bm25}(s,l) = \text{BM25}(s,l)$,$\phi_{cosine}(s,l) = \text{COSINE}(s,l)$, and $\phi_{bigram}(s,l) = \text{BIGRAM}(s,l)$ to compute the similarity between a short and a long text.

7.2.3.2 Topic Model-based Features

Topic models such as LDA [17] have been successful in discovering hidden topic distributions in a text corpus. Earlier works [120] have shown that matching texts based on the similarity models with hidden topics yield better results. Based on these findings, we define a set of topic model-related functions to compute the similarity between a short and a long text. To discover the topics, we run LDA on the articles in the corpus $\mathcal{C}$. Let T_i be the i^{th}topic discovered by the topic model and V_i be the set of top words in T_i. Let $\text{tf}(w;l)$ be the term frequency of term w in long text l . We define the following topic model based feature functions:

- **Topic Score** is defined as:

$$\phi_{T_i}(s,l) = \frac{1}{|l|}\sum_{w \in V_i} \text{tf}(w;l)$$

  It measures the relevance of a long text l to the topic T_i. Although this feature is not a function of short text s, it helps to measure the relevance of a long text l in the topic space of the corpus, which by construction includes the topic space of the short texts. For each topic T_i, one such similarity function is created.

- **Differential Topic Score** is defined as

$$\phi_{dt_i}(s,l) = \frac{1}{|l|} \sum_{w \in V_i \setminus s} \text{tf}(w;l)$$

It measures the similarity of a long text l to the topic words without considering the words in the short text s. The intuition here is that, we want to eliminate the part of the score that comes from matching words between short text s and long text l. Since IR-based feature functions already capture that, we want to get the score that solely comes from matching the long text to the topic T_i. Note that for every topic T_i, we define one feature function ϕ_{dt_i}.

Note that for the topic model based functions to be effective, it is very important to have a corpus with a topic distribution that represents the topic distribution of the short texts. More discussion on this is deferred to Section 7.3.3

7.2.3.3 Embedding-based Features

Word embeddings [99, 106] are vector representations of terms, and are computed from unlabeled data, that represent terms in a semantic space where the proximity of vectors can be interpreted as a semantic similarity. Word embed dings have been shown to produce good results in many works [76, 40] in the comparing of semantic similarities between terms, sentences, paragraphs, and documents. Inspired by these results, we define a bunch of feature functions that are based on the publicly available, pre-trained word vectors that are known as word2vec [99]. In the following sections, we use $v(w)$ to represent a low dimension vector representation of word w in the word2vec setting.

- **Word2Vec Score** is defined as

$$\phi_{w2v}(s,l) = \frac{1}{|s||l|} \sum_{w \in s} \sum_{w' \in l} \left(\text{tf}(w;s) \times \text{tf}(w;l) \times \max\left\{0, v(w) \bullet v\left(w'\right)\right\}\right)$$

It measures the term-level similarity between a short text s and a long text l by incorporating semantic similarity that is measured by the distance between the word vectors for those terms.

- **Word2Vec Topic Score** is defined as

$$\phi_{w2vt_i}(s,l) = \frac{1}{|l||V_i|}\sum_{w\in l}\sum_{w'\in V_i}\left(\text{tf}(w;l)\times\max\left\{0, v^T(w)\bullet v\left(w'\right)\right\}\right)$$

  This function is an extension of the "Topic Score" function to the semantic space. Here we measure the relevance of a long text l to the topic T_i by comparing the term-level semantic similarities that are measured by the distance between the word vectors for those terms.

- **Word2Vec Differential Topic Score** is defined as

$$\phi_{w2vdt_i}(s,l) = \frac{1}{|l||V_i \setminus s|}\sum_{w\in l}\sum_{w'\in V_i\setminus s}\left(\text{tf}(w;l)\times\max\left\{0, v^T(w)\bullet v\left(w'\right)\right\}\right)$$

  This is an extension of the "Differential Topic Score" function to the semantic space. It measures the semantic similarity of a long text l to the topic T_i through the word vectors, without considering the terms in the short text s.

7.2.3.4 Selective Expansion via the Bias Feature

Our observation shows that not all short texts need to be expanded to improve the accuracy of the task. Forcing expansion on all short texts sometimes reduces the task accuracy. In order to enforce selective expansion, we introduce a *Bias* feature. The bias feature is always set to -1

$$\phi_{bias}(s,l) = -1$$

A short text is expanded only if

$$\sum_f w_f\phi_f(s,l) + w_{bias}\phi_{bias}(s,l) \geq 0$$

That means, similarity score between short and long text has to be above some threshold to force mapping/expansion

$$\sum_f w_f\phi_f(s,l) \geq w_{bias}$$

7.2.4 Task-specific Learning of Mixture Weights w_f

Feature weights w_f are learned for a given task such that expanding short texts using the mappings obtained from w_f best improves the task accuracy. Let $\mathcal{L}\left(\mathcal{T}\left(\mathcal{D}_e\right)\right)$ be the loss function that is defined for task $\mathcal{T}$. It measures the empirical loss of the task on the expanded texts $\mathcal{D}_e$. By varying w_f, the mapping of short texts to long texts changes, which in turn changes the expansions $\mathcal{D}_s$, affecting the loss function. Our aim is to learn w_f such that the loss $\mathcal{L}$ is minimized for the task $\mathcal{T}$. Formally, we solve the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{w}}{\operatorname{argmin}} \mathcal{L}\left(\mathcal{T}\left(\mathcal{D}_e\right)\right) \\ =&\underset{\mathbf{w}}{\operatorname{argmin}} \mathcal{L}\left(\mathcal{T}\left(\left\{s_i \oplus \mathcal{M}_w\left(s_i\right)\right\}_{i=1}^{n}\right)\right) \end{aligned} \tag{7.3}$$

The form of loss function $\mathcal{L}$ is task dependent and unknown to us. It can be convex or non-convex; linear or non-linear; or can be extremely complex and non differentiable (for example, an output of a deep neural network.) Since our framework has to be task agnostic, we cannot make any assumptions about the form of loss function. Hence we use a derivative-free optimization technique that is known as BOBYQA [110]

BOBYQA solves bound constrained optimization problems without using derivatives of the objective function, which makes it a derivative-free algorithm. The algorithm solves the problem using a trust region method that forms quadratic approximation models of the objective function by interpolation. One new point is computed on each iteration, usually by solving a trust region sub problem subject to the bound constraints, or alternatively, by choosing a point to replace an interpolation point so as to promote good linear independence in the interpolation conditions. BOBYQA constructs the quadratic approximation models by the least Frobenius norm updating technique.

For a non-convex loss function $\mathcal{L}$, BOBYQA finds a solution that is at the local minimum. One of the standard techniques used to overcome the local minimum problem is to adopt random restarts. That is, we start with a random assignment of the weights w_f and run the BOBYQA procedure with that assignment as the starting value. By repeating this procedure multiple times with a different random initialization each time, and picking a solution that produces the least value for $\mathcal{L}$, we can possibly avoid a local minimum and achieve better solutions.

7.2.5 Alternate Minimization for Task-specific Learning

In this section we throw some insights into the learning that takes place with BOBYQA. Solving Equation 7.3 with BOBYQA involves evaluating the task $\mathcal{T}$ on the given set of expanded documents. In particular, evaluation of $\mathcal{L}$ involves fitting of the task parameters to the data first and then comparing the task results with the ground truth. It is important to distinguish the task-specific parameters from our model parameters. Let Θ be the parameters of the task (for example, in a classification task, Θ is word/feature weights, and in a clustering task Θ is cluster membership.) and W be our model parameters for the expanding of the short texts. The key idea in using BOBYQA is to learn Θ and W jointly, such that optimal task results are achieved through the optimal mapping of the short texts to the long texts. The procedure for this joint learning is outlined in Algorithm 9.

Considering a loss regularized framework for task $\mathcal{T}$ and the model parameters W of our framework, the optimal task parameters Θ^* are learned from the data (expanded texts) by optimizing the following objective shown in Step 4. Here $\mathcal{R}$ is the regularizer, and $\mathcal{L}_\mathcal{T}$ is the task-specific loss function (for example, hinge loss in SVM classification task). Note, $\mathcal{L}_\mathcal{T}$ is different from the loss function $\mathcal{L}$ that our framework uses to learn W. In the next step (Step 5), BOBYQA approximates the loss function $\mathcal{L}$ to a quadratic function $\mathcal{L}_Q$, using the interpolation technique. During this process, BOBYQA evaluates the loss function $\mathcal{L}$ at multiple points (W) using the task parameters Θ^*. In Step 6, the quadratic function $\mathcal{L}_Q$ is optimized to find the optimum parameters W^*. The updated weights W then produce new mappings/expansions for the short texts. The procedure repeats until no further update happens to W within the tolerance limit set in BOBYQA.

7.2.6 Practical BOBYQA: The Block Learning Approach

Minimizing the loss function $\mathcal{L}$ using BOBYQA results in multiple evaluations of the function $\mathcal{L}$ during quadratic approximation and the trust region growing/shrinking steps by BOBYQA. Each evaluation of $\mathcal{L}$ calls for the execution of task $\mathcal{T}$. For certain tasks, this evaluation can be time/resource intensive. For example, for a classification task, computing the classification loss for a given dataset involves training the classifier in the

Algorithm 9 Joint learning of our model parameters W and task parameters Θ

Require: Corpus $\mathcal{C}$, Short Texts Training Set $\mathcal{D}^{(\text{train})}$, Short Texts Development Set $\mathcal{D}^{(\text{dev})}$, Task $\mathcal{T}$

Ensure: Model parameters W

1: Randomly Initialize $W = [w_f]_{f=1}^{f=(\text{num features})}$

2: **while** not converged **do**

3: Find K candidate mapping long texts for every short text in $\mathcal{D}^{(\text{train})}$ and $\mathcal{D}^{(\text{dev})}$

$$\mathcal{M}_w(s_i) = \underset{l \in \left\{l_i^k \in \mathcal{C}\right\}_{k=1}^{K}}{\operatorname{argmax}} \sum_f w_f \phi_f(s_i, l)$$

4: Learn task parameters Θ

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \mathcal{R}(\Theta) + C \sum_{s_i \in \mathcal{D}^{(\text{train})}} \mathcal{L}_{\mathcal{T}}(e_i; \Theta, W)$$

where $e_i = s_i \oplus \mathcal{M}_w(s_i)$ is the feature vector of the expanded text

5: Quadratically approximate $\sum_{s_i \in \mathcal{D}^{(\text{dev})}} \mathcal{L}(e_i; \Theta^*, W)$ to $\mathcal{L}_Q\left(W; \mathcal{D}^{(\text{dev})}, \Theta^*\right)$. This approximation happens in BOBYQA.

6: Learn our model parameters W by minimizing the quadratic function $\mathcal{L}_Q$

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{\left|\mathcal{D}^{(\text{dev})}\right|} \mathcal{L}_Q\left(W; \mathcal{D}^{(\text{dev})}, \Theta^*\right)$$

7: **end while**

training split and then evaluating it on the test split. If the dataset size is very large, it may take a good amount of time to train the classifier. Hence, it becomes important to reduce the number of loss function evaluations.

The number of loss function evaluations depends upon the number of variables in the optimization problem, that is, the dimension of the weight vector W. In order to improve the learning (of weights w_f) time of the algorithm, we need to reduce the number of task evaluations, without reducing the number of features. In the following section we introduce a novel technique called "Block Learning" to achieve this.

In block learning, we divide our entire set of features into groups of smaller number of features called "blocks" and learn the weights one block at a time. Each block is a set of features of a particular class. For example, IR features constitute the 1st block, topic model features constitute the 2nd block and so on. The learning starts with block-1 where the BOBYQA finds optimum weights for the features in block-1. Then we move on to block-2, where BOBYQA learns weights for the features in block-2. At this stage, we treat block-1 as an additional feature and learn one weight for it. Next, we learn weights for features in block-3 using BOBYQA. Now, block-1 and block-2 together are treated as an additional feature and one weight is learned for it. The effective weights of each feature is the product of weights learned during its block and the additional feature weights learned during the subsequent blocks. This is depicted in Figure 7.2. The weight for feature f1 is $w_1 \times b_1 \times b_2$, where w_1 is the feature weight learned by BOBYQA during the first block; b_1 is the weight learned for the entire block-1 by treating block-1 as an additional feature; and b_2 is the weight learned for the entire block-1 and block-2 by treating block-1 and block-2 together as an additional feature. Similarly, weights for the other features are calculated.

The main advantage of block learning is that, at a time, BOBYQA has to optimize only those variables in a block along with one more variable for previous blocks. This reduces the number of evaluations of loss function $\mathcal{L}$ that BOBYQA makes, speeding up the learning process. Though block learning is not equivalent to learning all weights together, empirically, we observe that the task accuracy with expansion using block learned weights is at par with the accuracy from the expansion that uses weights learned with all features together. However, the total number of BOBYQA evaluations are significantly reduced

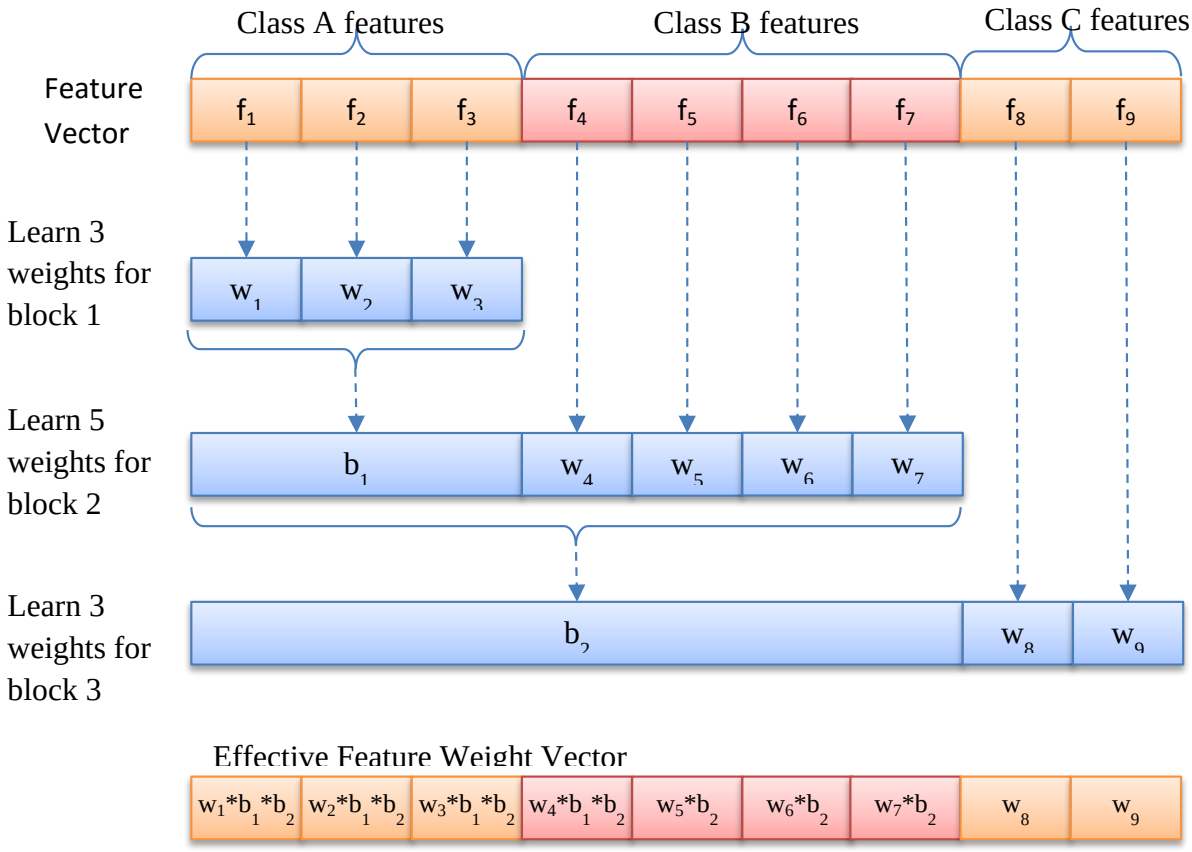


Figure 7.2: Block-learning approach.

with block learning, thus reducing the learning time.

7.3 Experiments and Evaluations

In order to evaluate our approach, we compare our work with several baselines and earlier works in the literature, which describe the handling of short texts. We demonstrate the effectiveness of our approach on two different ML tasks: classification and clustering. Our experimental results show that the proposed approach produces results that are comparable to the state-of-the-art techniques and that it is generic enough for application to many ML tasks. We have named our approach as TIDE (**T**ask-spec**I**fic short **D**ocument **E**xpansion), which has been used throughout these experiments and evaluations.

Note that we demonstrate the effectiveness of our approach on classification and clustering tasks. Although classification and clustering are not the goals of this thesis, as mentioned in the introduction, our approach is applicable to any IR/ML task, includ-

ing document organization. However, classification and clustering tasks have benchmark datasets and other methods/baselines to compare against, making them a suitable choice for the evaluation.

7.3.1 Short Text Tasks

In this chapter, we evaluate classification and clustering tasks for short texts. Though our technique can be applied to other types of ML tasks, we find that classification and clustering tasks on short texts have a good presence in the literature, which gives us an opportunity to use the standard data sets and compare against the baselines and earlier works. In these tasks, we represent short and long texts as TF-IDF vectors. To expand a short text s_i using a long text l_i, we smooth the TF-IDF vector of the short text using that of the long text in the following manner: $e_i = \alpha s_i + (1 - \alpha)\, l_i$ The $0 \leq \alpha \leq 1$ is a smoothing parameter that controls how much importance has to be given to short text and long texts. When $\alpha = 1$, only the short text used; when $\alpha = 0$, only the long text is used. The value of α is learned as part of the optimization in Equation 7.3. That is, we optimize the following to learn $\mathbf{w}$ and α:

$$\begin{aligned} &\underset{\mathbf{w},\alpha}{\text{argmin}} \mathcal{L}\left(\mathcal{T}\left(\mathcal{D}_e\right)\right) \\ =&\underset{\mathbf{w},\alpha}{\text{argmin}} \mathcal{L}\left(\mathcal{T}\left(\left\{\alpha s_i + (1-\alpha)\, \mathcal{M}_w\left(s_i\right)\right\}_{i=1}^{n}\right)\right) \end{aligned}$$

Using BOBYQA and the block learning machinery that is explained in Section 7.2.4, the optimization problem above is solved for $\mathbf{w}$ and α.

7.3.2 Datasets

We report our experimental results on several benchmark datasets used in the literature for classification and clustering tasks. We give an overview of the datasets, below.

7.3.2.1 Reuters21578

Reuters21578 dataset [115] is a collection of news articles. Each article has a short title and a long description of the news. All articles are classified into various topics. We take

the articles classified as "Corn" or "Wheat" and consider their titles as short texts. The task here is to classify the short titles into the Corn or Wheat class. The choice of these two classes is due to their high inter-class confusion while classifying short titles. We use the long descriptions (without the title in it) from the articles to form the universal corpus $\mathcal{C}$. The corpus is then indexed using Lucene software. Using Lucene as an IR system, we retrieve the top K candidate long texts by using the title as a short text query. The weights w_f are learned in order to choose a mapping long text for every short text such that expanding the short text by using the long text improves the classification accuracy.

In this dataset, we know the *true* long text for a short text (title), which is the long description of that news article. Using this true mapping we can compute a best classifier using the *true* long texts of the short texts. This helps us to compare our technique against the *true* expansion.

7.3.2.2 News Dataset from TagMyNews

TagMyNews[4] datasets is a collection of datasets of short text fragments that are used for the evaluation of the topic-based text classifier. One of the dataset from this collection is the News dataset. This is a dataset of ~ approximately 32K English news articles that is extracted from RSS feeds of popular newspaper websites (nyt.com, usatoday.com, reuters.com). Each news snippet has a title and a very short (one or two lines) description of the news event. Every news snippet is classified into one of the following categories: Sport, Business, U.S., Health, Sci&Tech, World and Entertainment. We use the titles and the short descriptions as the short texts for the classification task.

7.3.2.3 Web-Snippets

Web-Snippets [109] dataset has around 12K short web search snippets that are classified into seven classes. Out of this, 10K short texts are used to train the classifier and 2K for testing.

[4] `http://acube.di.unipi.it/tmn-dataset/`

7.3.2.4 ODPTweets

ODPtweets[5] is a large-scale Twitter dataset with nearly 25 million tweets that are categorized in the structure of the Open Directory Project (ODP). This dataset was used for the tweets classification task in WWW by Zubiaga et al. [166]. The categorization of tweets is inferred from the links that they point to. From the ODP style category structure that is associated with each tweet, we extracted the top-level ODP category as the category of the tweet. For example, for one of the tweets, the ODP category structure associated is "Computer/Software/Programming/Java". The top-level category is "Computer" in this case, which we associate with the tweet as the true category. There are 15 top-level categories in this dataset (Computer, Health, and the like). For each of these categories, we extracted around 600 tweets using Twitter API. We then discarded tweets that contained only junk characters or less than three words or non English tweets. This gave us as a collection with around 500 tweets in each category.

7.3.2.5 StackOverflowQuestions

We use the challenge data published in kaggle.com[6] [153] that contains questions that are posted by the users on stackoverflow.com[7]. This dataset consists of 3,370,528 questions posted on stackoverflow.com from July 31, 2012 to August 14, 2012. In our experiments, we randomly select 20,000 question titles as short texts from 20 different tags, as done by Xu et al. [153].

7.3.3 Corpus

Choosing the right universal corpus is very important. First, the universal corpus, as its name implies, must be large and rich enough to cover a lot of words, concepts, and topics that are relevant to the task problem. Second, this corpus should be consistent with the training and future unseen data that the task will deal with. This means that the nature of universal data (for example, patterns, statistics, and their co-occurrence of them) should

[5] `http://www.zubiaga.org/datasets/odptweets/`

[6] `http://www.kaggle.com/`

[7] `http://www.stackoverflow.com/`

be observed by humans to determine whether or not the potential topics analyzed from this data can help in making the task more robust. For example, the universal corpus has to help make the classifier more discriminate.

Today, Wikipedia is known as the richest online encyclopedia written collaboratively by a large number of contributors around the world. A huge number of documents are available in various languages and are placed in a organized structure which inspires the WWW, IR, and NLP research communities to use it as a large corpus [44].

Another data collection that can be used as a universal corpus in the medical domain is Ohsumed/MEDLINE. Unlike Wikipedia, Ohsumed only includes medical abstracts. This corpus can be used for tasks in the medical domain.

We use Wikipedia articles as a universal corpus $\mathcal{C}$ of long texts for experiments with TagMyNews, Web-snippets, ODPTweets and Stack Over flow Questions datasets. We use the Lucene software to index these articles and retrieve the top K candidate long texts using short texts as queries. The weights w_f are learned such that expanding the short texts using the long texts (Wikipedia articles) improves the classification and clustering accuracy.

7.3.4 Robustness Analysis of Our Approach

One of the important characteristics of our approach is its robustness in the noise in the corpus. Unlike other approaches [20, 121, 157] our method does not force expansion on all the short texts. Expansion is done only if that helps improving the performance of the task. When a different corpus (one from a different domain) or a noisy corpus is used, it may not provide the right long texts for the expansion and, hence, expansion may not boost the performance of the task. In our experiments, we demonstrate the robustness of our method against the noisy corpus from the following three scenarios: (i) a wrong corpus (ii) a corpus that is partially relevant or has noise and (iii) a corpus that contains the short texts themselves. In the third scenario, we show that our method is capable of avoiding the selection of short texts from the corpus for expansion, whereas, other IR-based expansion techniques result in the selection of short texts themselves from the corpus for the expansion, which does not help the task to improve the performance.

Expt#	Dataset	Short Text	Corpus
1	Reuters21578	News Title	Articles from entire Reuters21578 collection
2	Web-Snippets	Snippet	Wikipedia articles
3	TagMyNews	Title + RSS feed	Wikipedia articles
4	ODPTweets	Tweet	Wikipedia articles
5	TagMyNews	Title + RSS feed	Wikipedia articles + all short texts from the dataset
6	TagMyNews	Title + RSS feed	Wikipedia articles + Ohsumed
7	TagMyNews	Title + RSS feed	Ohsumed

Table 7.1: Experiments for the classification task using different datasets and corpora

7.3.5 Evaluation Methodology

All the ML tasks are carried out by expanding the short texts and by measuring the improvement in the task's performance. We compare our expansion technique against various baselines and previous works. The two baselines we compare against are (i) IR system-based expansion and (ii) Word2Vec-based expansion

In an IR system based expansion, the long text articles in the corpus are initially indexed. Using the short text as a query, the top ranked result from the IR system is used to expand the short text. In our experiments, we used Lucene as the retrieval system.

In Word2Vec based expansion, a word vector for every word in the short text is obtained using the word2vec tool. The average word vector is then computed from all these word vectors and appended to the short text to produce a long/expanded text.

7.3.5.1 Methodology for Classification Task

There are various criteria that can determine the effectiveness of a classification task; however, precision, recall, and accuracy are most often used. We choose accuracy (macro accuracy in case of multi-class classification) to measure the performance of the task, though other criteria may equally be used. In fact, it does not matter which criteria

Expt#	Accuracy (%)						% Short
	Short Text Only	Lucene First	Word2Vec	TIDE	Comparing Technique	Actual	Texts Not Expanded
1	85.8	87.2	67.5	91.6	-	92.1	14
2	62.1	76.8	52.9	84.2	82.2 (Phan [109]) 85.3 (Chen [34])	-	8
3	71.3	73.7	56.3	81.3	80 (Vitale [145])	-	11
4	39.4	41.3	21.2	47.8	-	-	0
5	71.3	71.8	56.3	81.1	-	-	0
6	71.3	73.1	56.3	81	-	-	10
7	71.3	62.3	56.3	70.1	-	-	92

Table 7.2: Accuracy comparison of classification task (Note, TIDE is our approach)

we choose because the goal of our experiments is to demonstrate an improvement in the task's performance using our technique and not to judge the task itself.

The datasets described in Section 7.3.2 are divided into train and test splits according to the previous works using those datasets. We use 25% of the data from the test split as the development set and the remaining as the test set for evaluation. During the training phase, the model parameters (of both our framework and classier) are optimized by training the classifier on the expanded short texts from the training set and measuring the accuracy on the expanded short texts from the development set. In the testing phase, expanded short texts from the test set are classified and accuracy is measured.

We report the results for the task using the SVM classification algorithm, however, we observed a similar behavior when using other classification algorithms.

7.3.5.2 Methodology for Clustering Task

We test our algorithm on the Stack Over flow Questions dataset. The clustering performance is evaluated by comparing the clustering results of short texts with the tags/labels provided by the text corpus. The accuracy [26] and NMI metrics[92] are used to evaluate the clusters.

Since the focus our investigation is to demonstrate the improvement in the clustering accuracy through our expansion technique rather than the clustering method, we used the standard k-means algorithm (which, is also the algorithm used in the work we compare against.)

7.3.6 Results and Discussion

Next we present and discuss our results on classification and clustering tasks, and also compare them with baselines and other methods.

7.3.6.1 Classification Accuracy

To investigate the accuracy improvement of a short text task using our approach, we have designed several experiments, as shown in Table 7.1. For each of these experiments Table 7.1 shows the dataset, the short texts, and the universal corpus used. Experiments 1-4 show expansion using the right corpus and 5-7 show the robustness of our approach against the incorrect or noisy corpus.

Table 7.2 shows classification accuracies of our method against other baselines. In the case of Reuters21578 (Experiment 1) we use Reuters21578 articles as the corpus. That means, the true expansion of the short text (title) is present in the corpus. We consider the body of a news article as the true expansion of the news title. This experiment lets us investigate how close our approach can perform to the true (oracle based) expansion. Interestingly, we observe that our method achieves 91.6% accuracy which is close to the true expansion accuracy of 92.1%.

In Experiment 5, the corpus contains exact short texts from the dataset as short articles. The IR method retrieves these short texts as top results due to the high matching score and, hence, does not help the expansion. Experiment 6 has the corpus with the noise: Wikipedia articles mixed with Ohsumed articles. We observe that the Wikipedia articles provide the right candidates for expansion, whereas Ohsumed articles are irrelevant (noise) for the news snippets in the TagMyNews dataset. The IR method does a good job of selecting only relevant candidates for the expansion from the corpus and discards the noisy candidates. In addition, our model assists in the selecting of expansion texts, which improves the task accuracy. Experiment 7 has an irrelevant corpus for the expansion. As we can see from the results, Lucene First (an IR method) is forced to choose a best-matching candidate for the expansion. Meanwhile, forcing this expansion reduces the task accuracy. However, our model's selective expansion strategy makes the short texts not expand in such scenarios. As shown in Table 7.2, the percentage of short texts not expanded is up to 92% in this case.

Table 7.2 also shows comparison of classification accuracies reported in short text classification techniques from the literature with our method. In most of the cases, our method performs at par with other techniques or beats them. In comparison to [34], we perform slightly worse. Note that, we carried our experiments on the same datasets published by Chen et al. [34]. We believe that the drop in classification accuracy in our system is due to the differences between the corpus used in our method and that used by Chen et al. [34]. The seed words that are used for crawling the Wikipedia and the process of building the corpus are not clear from the descriptions given by Chen et al. [34]. We used all the Wikipedia articles in our experiments; however, we believe that the accuracy of our method would improve if we build a focused crawler to generate more relevant corpus.

7.3.6.2 Clustering Accuracy and NMI

The results of the clustering task are shown in Table 7.3. We observe that our method beats all the baselines and performs at par with Xu et al. [153]. The Lucene First and Word2Vec-based expansions do not consider the clustering accuracy and NMI during expansion, our method, however does. This leads to better task performance when compared to these baselines.

Baselines/Comparing techniques	Accuracy(%)	NMI(%)
Short Text Only	26.3	30.2
Lucene First	38.8	40.1
Word2Vec	11.4	13.6
Jiaming Xu et al. [153]	51.1	49
TIDE	50.8	52.4

Table 7.3: Accuracy and NMI comparison of clustering task (TIDE is our approach)

Note that the framework and features used for the clustering task is same as that of the classification task. Hence, we state that our framework is task agnostic. However, it learns to expand the short texts for the classification and the clustering tasks in a way that improves the task's performance. This makes our framework powerful for adoption by many IR/ML/NLP tasks that deal with short texts.

7.3.6.3 Effect of Block Learning

In the next set of experiments, we investigate the effect of the block learning mechanism. For the classification task, we run the experiments with and without block learning in order to investigate (i) the amount of time that block learning saves and (ii) the effect on classification accuracy. Figure 7.3 shows that in some cases there are 30% lesser task evaluations with block learning, with a marginal drop in the classification accuracy. Note that we report the saving in training time w.r.t. the number of task evaluations, because the absolute time for a task depends upon various factors such as the size of the dataset, the task training/validation methodology, and the like. Interestingly, in the case of TagMyNews, we observe a slight increase in the classification accuracy with block learning. We believe that this is because of our model settles for a local minimum in a high-dimensional feature space when all the features are used together as one block (no block learning). Meanwhile, when block learning is employed, the reduction in the feature space dimension helps to find better solutions.

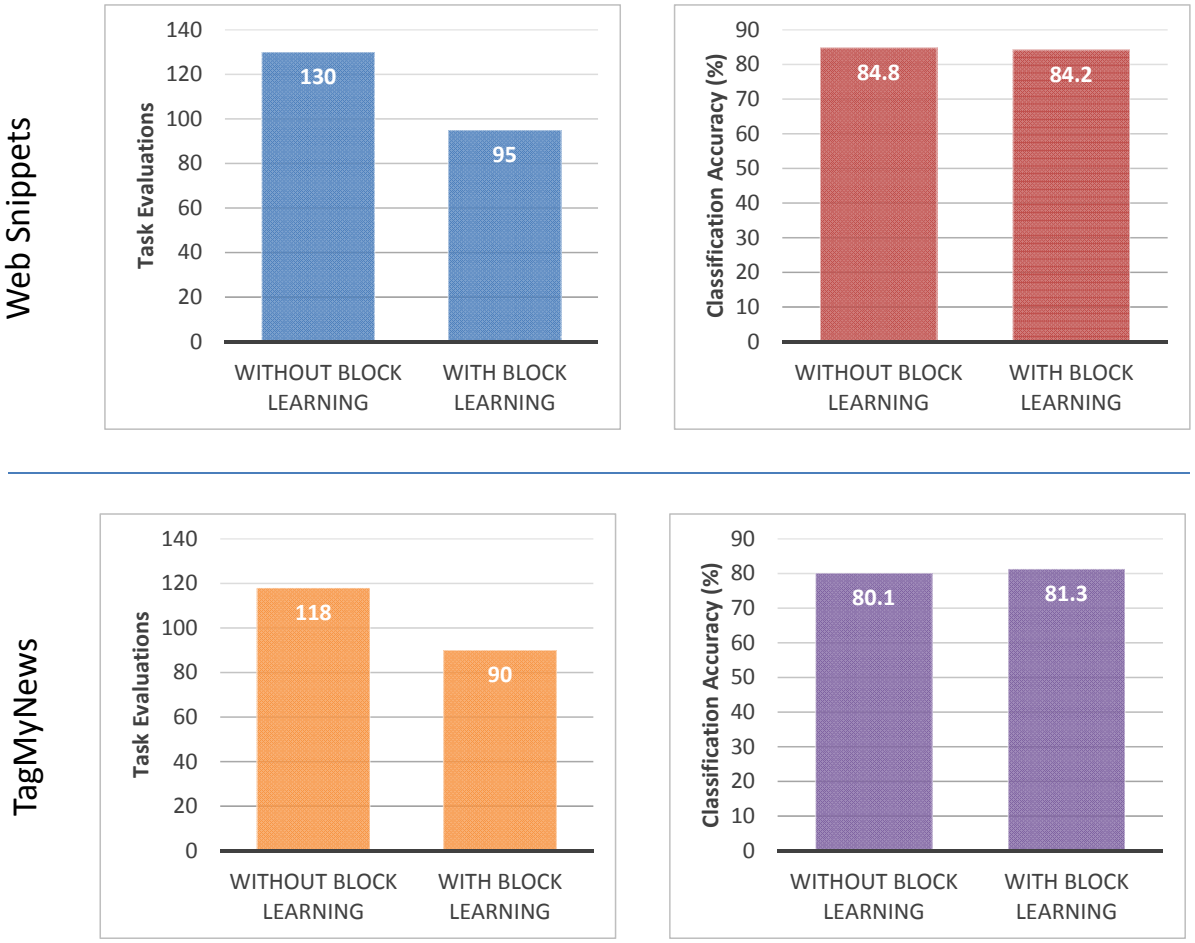


Figure 7.3: Block Learning: Reduction in the number of task evaluations and impact on accuracy

7.3.6.4 Feature Ablation

In this section we investigate the usefulness of different classes of feature functions through feature ablation tests. We start with only IR features and then incorporate topic model and embedding features one by one. We then compare how the learning of expansion improves the classification task accuracy with the addition of these features. Figure 7.4 shows the improvement in classification ac curacies as we add different classes of features. IR based features alone are able to achieve a gain of around 5% in accuracy over Lucene first, followed by another 4 to 5% gain through topic model-based features. While IR features can be computed very easily, topic model features (TM) require a one time computation of the topic distribution of the corpus from LDA or similar mechanisms. There is a marginal improvement when word2vec features (W2V) are used. In these experiments,

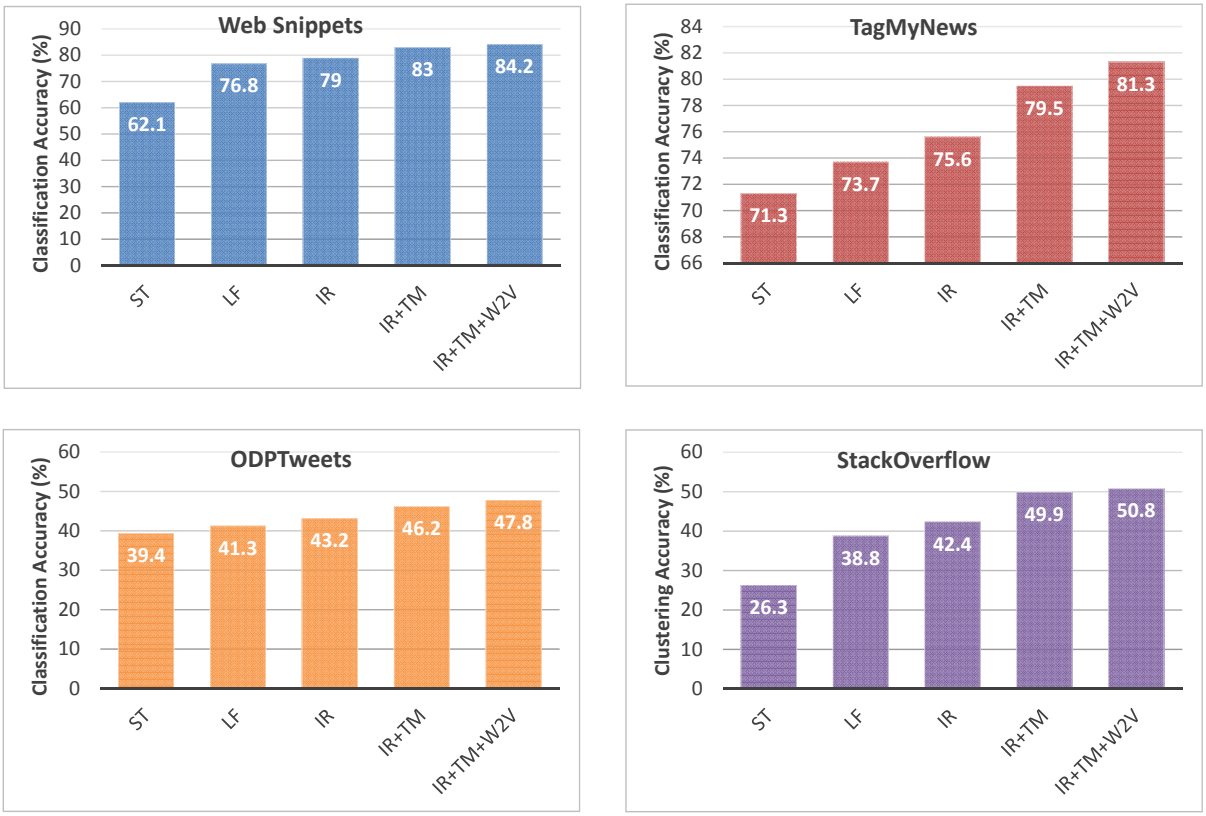


Figure 7.4: Feature Ablation Results

we use pre-built word vectors (of 300 dimension) published by Google. Experiments using word vectors trained from the corpus and of different dimensions will be part of our future work.

The choice of adding IR features, followed by TM features, and then W2V features comes from our observation that IR+TM features produced better task accuracy than IR+W2V features. However, when all the tree classes of features were combined, we observed the best improvement in the accuracy.

7.3.6.5 Random Restart Results

To overcome the problem of the local minimum with our approach, we adopt a standard technique of random restarts. In this section we investigate the effect of random restarts on the classification accuracy of the task by plotting the best classification accuracy that is observed so far against the number of random restarts. Figure 7.5 shows the improvement in the accuracy over multiple random restarts. We observe that in approximately 65 to 75

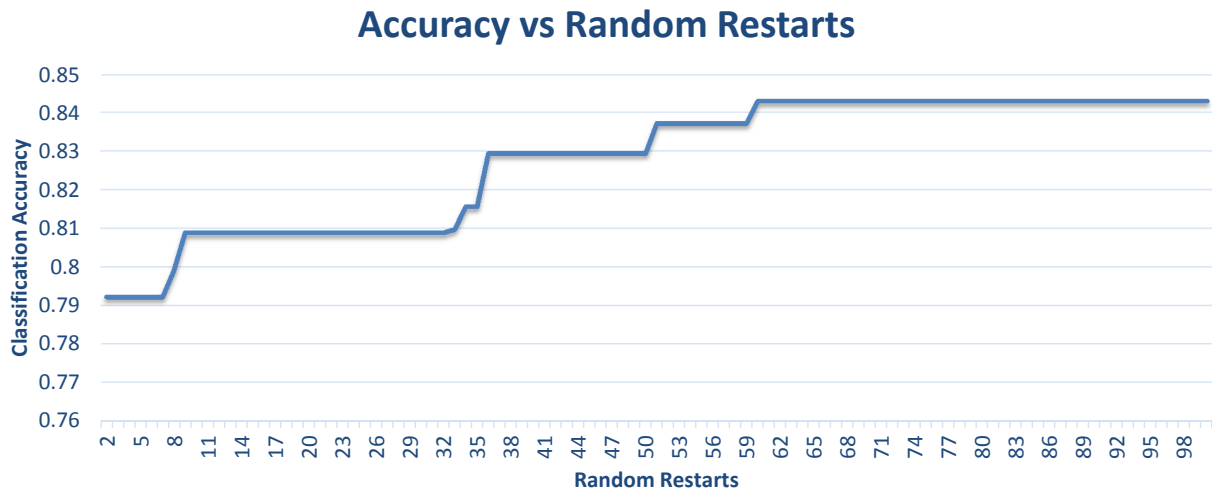


Figure 7.5: Classification Accuracy improvement with Random Restarts in experiments using Web-snippets dataset (Experiment 2)

random restarts, we reach the maximum task improvement that can be achieved by our method.

7.4 Conclusion

In this work we presented a technique for learning to expand short texts in a task-specific way. The expansion is such that the task accuracy best improves when expanded texts are used. We do not make any assumptions regarding the tasks except that the task can be evaluated with the expanded texts. Hence, our technique can be adopted to expand short texts for any task. To learn task-specific expansion, we presented several classes of mapping feature functions: IR, topic model and embedding-based. Using a derivative-free optimization technique known as BOBYQA, we presented the efficient learning of feature weights in a block learning fashion.

Chapter 8

Conclusion

In this thesis we studied the problem of evolving a hierarchy of categories that best organize the documents in a collection under soft (preferential) and hard (necessary) constraints, using a massive knowledge graph. The evolved categorization system addressed four important challenges: (i) under-specified organization, (ii) over-specified organization, (iii) intent coverage, and (iv) temporal relevance.

Under-specified categories can result when an insufficient number of categories is used to describe the documents in the collection. For example, the organization the of Reuters news articles into the category structure provided by 20NewsGroup results in an under-specified organization. Our experiment to classify Reuters RCV-v1 using 20NG categories resulted in an highly skewed distribution of the documents over the categories. Some of the categories were assigned very few documents and others too many. This happened due to very small number of categories being relevant to the document collection.

We addressed the problem of under-specified categorization by associating fine-grained categories with every document. In order to achieve this, we proposed a solution by adopting a global, collaboratively developed knowledge graph of categories (such as Wikipedia) to a local document categorization problem effectively. We presented a principled approach to evolve an organization-specific, multi-class, multi-label document classification system starting from scratch (Chapter 2). We modeled our classification problem as that of inferring structured labels in an Associative Markov Network (AMN), where the label space was derived from a global knowledge graph of categories. By incorporating the outputs of various classification techniques (such as SVMs, Topic Models, Knowledge Mining,

and the like) as dynamic features in AMN along with the document similarity measures (such as Jaccard, Cosine, BM25, and the like) as static features, we presented a novel framework for learning to associate categories with the documents.

Associating fine-grained categories with every document in a collection posed another challenge: over-specified categorization. When the number of categories are too many for the document collection, then each document is assigned in its own category leading to very sparse and less interpretable categorization.

In order to overcome the over-specified categorization, we proposed a novel technique of summarizing DAG-structured category hierarchies over a given set of documents (Chapter 4.) The summarization produced a smaller set of categories that are still representative of the documents in the collection. We posed the problem as a submodular optimization problem on a category hierarchy using the documents as features. We proposed many desirable properties for the chosen categories which included document coverage, specificity, category diversity, and category homogeneity, each of which, we showed can be modeled as a submodular function. We showed that other information provided, for example by unsupervised approaches such as LDA and its variants, could also be utilized by defining a submodular function that expressed coherence between the chosen topics and this information. We used a large-margin framework to learn convex mixtures over the set of submodular functions. Using greedy inference procedure we showed how to generate a summary category set describing the document collection.

In addressing over-specified organization, we also presented ideas for unsupervised techniques using Sampling and Expectation Maximization, for identifying a smaller DAG structured categories for organizing the document collection (Chapter 6.) Given a collection of documents with fine-grained concept assignment, and a massive concept hierarchy in the form of a DAG, we inferred a sub-DAG as a summary DAG that best represented the collection. We modeled the category DAG as a Markov Network for generating the documents in the collection and specified a joint probability distribution over the observed documents in the collection. Through Gibbs sampling we learned this distribution. Thereafter we estimated the importance of categories based on the marginal probabilities of the category nodes and their children in order to rank the categories. By taking top K ranked nodes along with the edges between them, we formed a sub-DAG.

In order to address the third challenge of document organization, the intent coverage, we proposed personalization techniques for associating categories with documents (Chapters 1 and 2.) Personalization is done by seeking user feedback on the system generated document-category pairs. The feedback is then translated into hard and soft constraints which are applied during AMN inference for category association with the documents. To reduce this cognitive load and to achieve greater accuracy with fewer training labels (that is, feedback) we presented a novel joint Active Learning technique, in order to jointly select document and category pairs for feedback. We showed that our technique is able to learn to categorize documents with fewer feedbacks/labels thus reducing the cognitive load on the users.

The final challenge, the temporal relevance, is naturally addressed in our techniques due to the usage of collaboratively developed Knowledge Bases (such as Wikipedia) as the vocabulary for the categories. Since the Knowledge Bases that we use in our framework are kept up-to- date with all the categories (by the collaborative editors,) we are able to spot new categories and associate them with the documents when new topics emerge in the document collection. We also proposed ideas for updating already categorized documents when new categories are added to the Knowledge Bases.

Apart from addressing these four challenges, we also presented a technique for diversifying the category assignments to the documents using knowledge graphs (Chapter 4.) This becomes important when there is a budget constraint on the number of categories that can be associated to a document. In order to produce a diverse but relevant set of categories in the precious top K positions, we presented a novel technique using a Biconvex optimization formulation which is a graph based iterative method for choosing diversified categories.

Another important contribution of this thesis is the organization of short texts (Chapter 7.) Short texts are quite different from the traditional documents due to their shortness and sparseness. Hence, our earlier developed methods could not be applied directly on short texts. We presented a method for expanding short texts using a "universal corpus" (such as Wikipedia, Ohsumed) of documents, for the better organization of short texts. Our method used a derivative free algorithm called BOBYQA to iteratively reduce the loss function for the task over the short texts. In each iteration of the algorithm we generated an expansion candidate and then evaluated the task with the candidate expansion text.

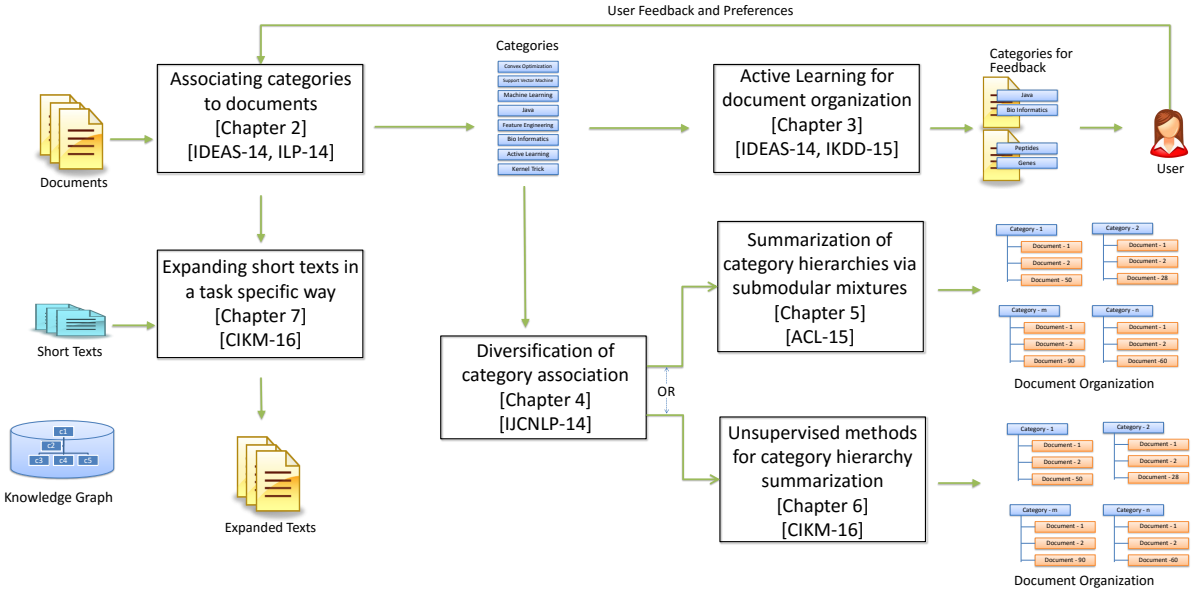


Figure 8.1: Summary of this thesis

Based on the result of the evaluation, we computed a quadratic approximation of the loss function and further minimized using a trust region technique. This made each iteration to improve the short text expansion optimally. As a result, we learned a model to expand the short text by mapping it to the best possible article from the "universal corpus".

Although we confined our experimental studies to organize digital documents in this thesis, the proposed techniques may be useful in other settings as well. For example, it will be an interesting study to apply our techniques for automatic table of content (ToC) suggestions; Wikipedia style disambiguation page generation and the like. In case of ToC generation, we can reformulate the problem as that of organizing chapters under summarized categories as ToC. Each chapter has a rich context (section headings, description text, and the like), which can be used to associate sections/chapters to various concepts from the Knowledge Base. Summarizing these concepts at different levels can generate ToC. Similar analogy can be made with other examples and cases. However, we leave these studies for future work.

In Figure 8.1 we summarize the organization of all these techniques and show how they are interconnected. This figure also shows the different conferences in which each of these sub-problem is published.

Finally, we developed an end-to-end system by piecing together all of these sub-systems

and built a categorization system . For this, we extracted about 150 technical articles under Science tracks (Computer Science, Chemistry, Computational Biology, Micro Biology, Genetics, Physics, Electricity, Logic (Mathematics), Algebra, and Number Theory) from DOAJ and arXiv collections. We associated categories for each article using our AMN framework (Chapter 2) with Wikipedia as the Knowledge Base. While associating concepts from Wikipedia to the articles, we also added user preferences (constraints) to limit the categories to only those under "Science" in Wikipedia. Using our active learning technique (Chapter 3) we provided some feedback to the system to improve the accuracy of the categorization. We then applied our diversification algorithm (Chapter 4) to the associated categories so that each document is associated with at most 20 categories. Thereafter, using our proposed submodular category summarization technique (Chapter 5), we summarized the set of categories down to 100 categories. The Figure 8.2 shows the summarized 100 categories in the form of a tag cloud. The font sizes of the category names indicate the number of documents organized under them.

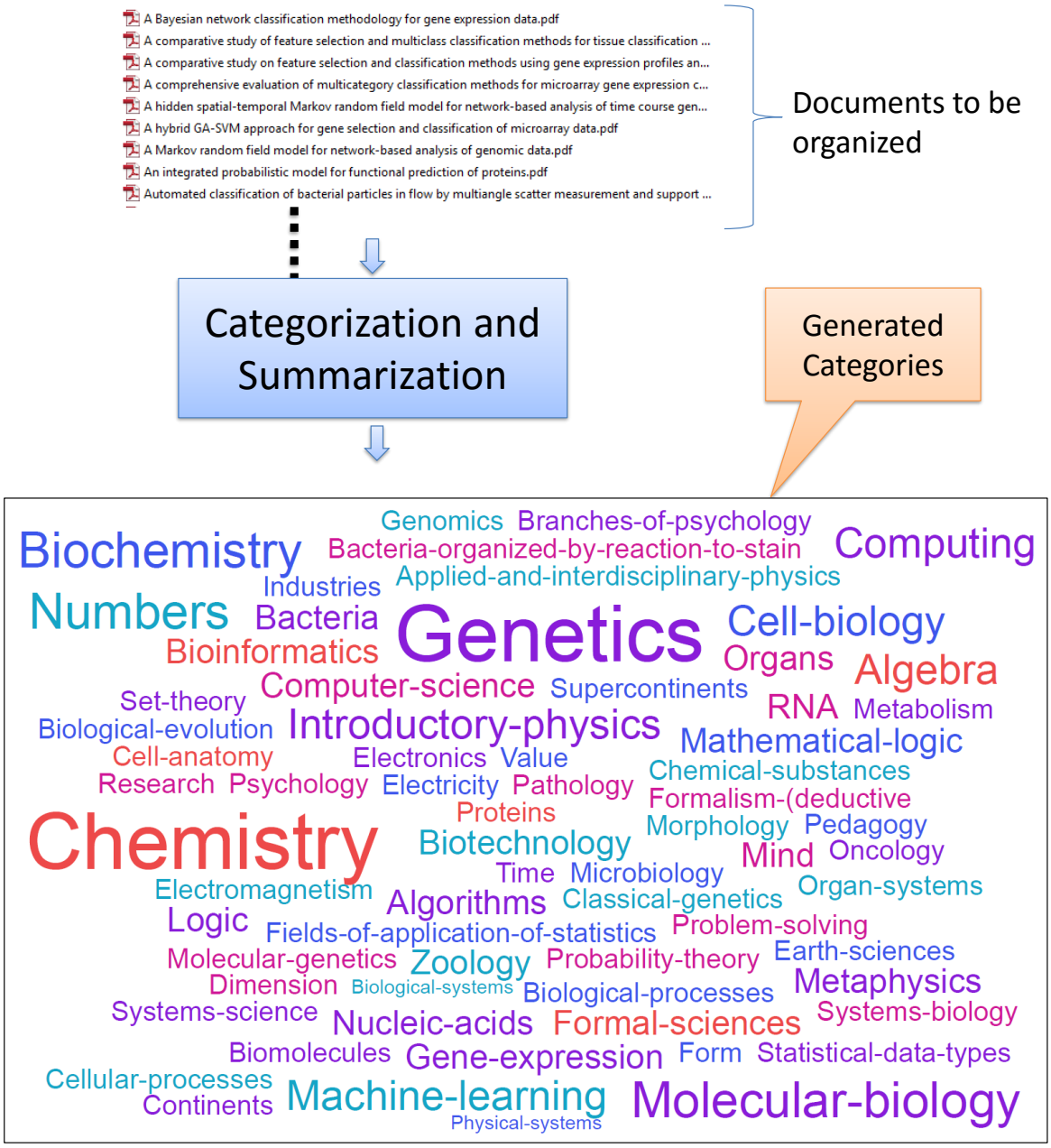


Figure 8.2: Categories generated by our End-to-End system on the sample document collection.

Appendix A

Cycle Detection and Breaking

The Wikipedia category hierarchy graph is not a DAG. There are several cycles in the category hierarchy. In order to perform Bayesian analysis (Chapter 6) and summarization (Chapter 5) on this hierarchy, we need to first convert it to a DAG. Our semantic interpretation of the hierarchy is that, in every path from root to leaf, a category higher up in the path is more general than the categories below it. For example, category "Science" is more general than category "Physics". Representing this relationship as a directed edge $A \rightarrow B$, where A is more general (an ancestor category) than B (a descendant category,) we say that there should not be any edge of the form $B \rightarrow A$ on the entire path. Programmatically verifying if A is more general than B for any two given categories lying on a path and appropriately ordering them for eliminating the cycles is a research problem by itself. A simpler heuristic to get rid of a cycle is to delete the very first edge on a path that connects a descendant category to an ancestor category forming a cycle. The Figure A.1 shows an example illustrating this procedure. The rationale behind choosing such an edge is that, a more specific category cannot be a parent (or ancestor) of less specific (equivalently, more generic) category.

The Algorithm 10 outlines our cycle detection and elimination procedure. It traverses all the paths starting at every node, checking if any of the paths end up in the starting node itself, hence forming a cycle. If such a path exists, the edge that returns to the starting node of the path is deleted to break the cycle. Since the algorithm traverses in a breadth first fashion, it guarantees that the very first edge on the path that connects a descendant category to an ancestor category gets dropped. On the Wikipedia (2012 Oct dump) category graph, our algorithm detected and eliminated 1766 cycles.

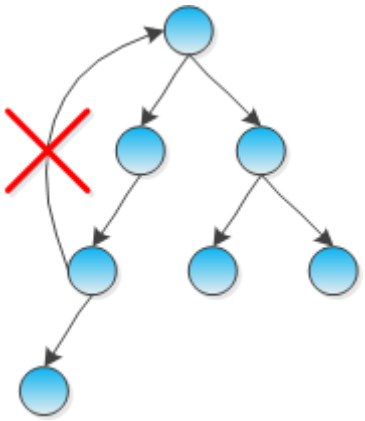


Figure A.1: Cycle breaking

Algorithm 10 Cycle Detection and Elimination

Input : Wikipedia Category Graph with Cycles

Output : Wikipedia Category DAG (without cycles)

Let $[C_1, C_2, ...C_n]$ be the ordered list of Wikipedia categories visited during the Breadth First traversal of Wikipedia categories starting from the root node.

for $i = 1..n$ **do**

    Let $[C_{i_1}, C_{i_2}, ...C_{i_n}]$ be the ordered list of Wikipedia categories visited during the Breadth First traversal of Wikipedia categories starting from C_i

    for $j = i_1..i_n$ **do**

        if $C_i \in \{\text{children of } C_j\}$ **then**

            Delete edge $C_j \rightarrow C_i$

        end if

    end for

end for

While other approaches such as finding minimum number of edges to eliminate all the cycles in the hierarchy, or employing NLP based techniques for finding/correcting the violations of "general→specific" ordering to eliminate the cycles are possible (which of course need further investigations,) we restrict our strategy to the above mentioned heuristic in this thesis.

Appendix B

Proof of Equation 6.19

The quantity $\alpha(j)$ is the probability of reaching node j from the root . This is evident from the definition in Equation 6.17

The quantity $\beta(j)$ in Equation 6.18 is the expected importance score that propagates through node j from all the leaf nodes. That is, if we propagate the importance scores $N_l \in [0, 1]$ from the leaf nodes upwards by dividing the scores proportional to the edge probabilities, the amount of score that reaches the node j through all the paths is given by $\beta(j)$.

Note that, in the numerator of Equation 6.19, all paths in $\mathcal{P}$ pass through the edge (j, k). Hence $\mathcal{P}$ can be re-written as follows:

$$\mathcal{P} = \mathcal{U} \otimes (j, k) \otimes \mathcal{V}$$

where $\mathcal{U}$ is the set of paths ending at node j and $\mathcal{V}$ is the set of edges starting at node k. The operator $\otimes$ is the cross join between the set of paths. For example, if $\mathcal{U}$ has two paths p_1 and p_2, and $\mathcal{V}$ has two paths q_1 and q_2, then $\mathcal{U} \times \mathcal{V}$ has four paths $\{p_1q_1, p_1q_2, p_2q_1, p_2q_2\}$.

Any function f over $\mathcal{P}$ can be decomposed over $\mathcal{U}$ and $\mathcal{V}$ as follows:

$$\sum_{p \in \mathcal{P}} \mathrm{f}(p) = \left(\sum_{u \in \mathcal{U}} \mathrm{f}(u) \right) \times \left(\sum_{v \in \mathcal{V}} \mathrm{f}(v) \right)$$

Applying this principle to the Equation 6.16, we can re-write it as follows:

$$p^{(t)}(k|j) = \frac{\sum_l N_l \frac{1}{Z_l} \left(p^{(t-1)}(l|\mathrm{end}(\mathcal{P})) \left(\sum_{\mathcal{U}:\mathrm{end}(\mathcal{U})=j} \prod_{(r,s)\in\mathcal{U}} p^{(t-1)}(s|r) \right) p^{(t-1)}(k|j) \left(\sum_{\mathcal{V}:\mathrm{start}(\mathcal{V})=k} \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right) \right)}{\sum_l N_l \frac{1}{Z_l} \left(p^{(t-1)}(l|\mathrm{end}(\mathcal{P})) \left(\sum_{\mathcal{U}:\mathrm{end}(\mathcal{U})=j} \prod_{(r,s)\in\mathcal{U}} p^{(t-1)}(s|r) \right) \left(\sum_{\mathcal{V}:\mathrm{start}(\mathcal{V})=j} \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right) \right)}$$

Bringing the part which is independent of l outside the summation, we get:

$$p^{(t)}(k|j) = \frac{\left(\sum_{\mathcal{U}:\text{end}(\mathcal{U})=j^{(r,s)}\in\mathcal{U}} \prod p^{(t-1)}(s|r)\right) p^{(t-1)}(k|j) \left(\sum_l N_l \frac{1}{Z_l} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \left(\sum_{\mathcal{V}:\text{start}(\mathcal{V})=k(r,s)\in\mathcal{P}} \prod p^{(t-1)}(s|r)\right)\right)\right)}{\left(\sum_{\mathcal{U}:\text{end}(\mathcal{U})=j^{(r,s)}\in\mathcal{U}} \prod p^{(t-1)}(s|r)\right) \left(\sum_l N_l \frac{1}{Z_l} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \left(\sum_{\mathcal{V}:\text{start}(\mathcal{V})=j(r,s)\in\mathcal{P}} \prod p^{(t-1)}(s|r)\right)\right)\right)}$$

Note, $Z_l = \alpha(l)$. Substituting this in the above equation, we get:

$$\begin{aligned} p^{(t)}(k|j) &= \frac{\left(\sum_{\mathcal{U}:\text{end}(\mathcal{U})=j^{(r,s)}\in\mathcal{U}} \prod p^{(t-1)}(s|r)\right) p^{(t-1)}(k|j) \left(\sum_l \frac{N_l}{\alpha(l)} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \left(\sum_{\mathcal{V}:\text{start}(\mathcal{V})=k(r,s)\in\mathcal{P}} \prod p^{(t-1)}(s|r)\right)\right)\right)}{\left(\sum_{\mathcal{U}:\text{end}(\mathcal{U})=j^{(r,s)}\in\mathcal{U}} \prod p^{(t-1)}(s|r)\right) \left(\sum_l \frac{N_l}{\alpha(l)} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \left(\sum_{\mathcal{V}:\text{start}(\mathcal{V})=j(r,s)\in\mathcal{P}} \prod p^{(t-1)}(s|r)\right)\right)\right)} \\ &= \frac{\alpha(j) p^{(t-1)}(k|j)\, \beta(k)}{\alpha(j)\beta(j)} \end{aligned}$$

□

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisors Dr. Ganesh Ramakrishnan and Dr. Mark Carman for their continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my PhD study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Soumen Chakrabarthy and Dr. Yuan-Fang Li for their insightful comments, assessments and encouragement.

My particular thanks goes to Rishabh Iyer, Jeff Bilmes and Vikas Sindhwani for their active collaboration with my research work. Although they were busy with their own research, they have always been available to answer my questions, clear by doubts and show directions.

My sincere thanks also goes to Raghavendra Udupa from Microsoft Research, India who provided me an opportunity to join their team as an intern, and who gave access to the laboratory and research facilities. Without his kind support it would not have been possible to complete this research.

I am thankful to Dr. Mohan Krishnamurthy and Dr. Sastry, CEOs of IITB-Monash Research Academy, for their support and advice during my research work. I thank all the members of Department of Computer Science, IIT Bombay and Caufield school of IT, Monash University, who created a friendly environment for research. My special thanks goes to the staff at IITB-Monash Research Academy, IIT Bombay, and Monash University. Some of the important names I would like to mention are: Mrs. Jayasree,

Mrs. Priyanaka, Mrs Laya, Mrs. Kuheli, Mr. Kiran, Mr. Rahul, Mr. Bharat, Mr. Vijay Ambre, Mrs. Sunanda, Mr. Thushar, Mrs. Alpana Athavankar and Mrs. Gaikwad. I am very thankful to Ms Sheba Sanjay, Communications and Training Services Consultant at IITB-Monash Research Academy, for her timely language reviews of my publications and this thesis.

I would never forget all the chats and beautiful moments I shared with many of my friends and classmates. They were fundamental in supporting me during these stressful and difficult moments. I would like to mention my thanks to: Ajay, Arun, Ashish, Uma, Chetana, Abhijit, Joe, Sudha, Vishwajeet, Ananth, Jinesh, Kanika, Satyabrata, Ambha and Upasana.

My special thanks goes to Srikrishnan, Aslam and Kashi from IBM, without their advice and recommendation I would not have started this journey.

I am extremely grateful to my parents Balakrishna Bairi and Vijayalakshmi for their love, prayers, caring and sacrifices for educating and preparing me for my future. A special thanks to my father-in-law Gopalakrishna and mother-in-law Swarnalatha for their support and well wishes during my research work. I also express my thanks to my brother Ananthakrishna for his support and help during my PhD. I am indebted to my wife Akshaya for her love, understanding, sacrifices, prayers and continuing support to complete this research work. My heartiest thanks to my son Dhanvin for giving me happiness with his innocent look, talk and cheerful play which took away all my stress.

Finally I thank my god for all his grace and blessings throughout my life.

Date: _____ Ramakrishna B Bairi

Bibliography

[1] 20Newsgroups. `http://qwone.com/~jason/20Newsgroups/`.

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[3] John Aitchison. *The statistical analysis of compositional data,*. Chapman and Hall London, 1986.

[4] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 3–12. IEEE, 2008.

[5] Mihael Ankerst, Christian Elsen, Martin Ester, and Hans-Peter Kriegel. Visual classification: An interactive approach to decision tree construction. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–396. ACM, 1999.

[6] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE, 2012.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[8] James K Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132, 1979.

[9] Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. Clustering short texts using wikipedia. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM, 2007.

[10] Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[11] Barry Becker, Ron Kohavi, and Dan Sommerfield. Visualizing the simple bayesian classifier. *Information visualization in data mining and knowledge discovery*, 18:237–249, 2001.

[12] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 33–44, New York, NY, USA, 2008. ACM.

[13] James Benhardus and Jugal Kalita. Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139, 2013.

[14] Wei Bi and James T Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17–24, 2011.

[15] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, volume 16, page 17. The MIT Press, 2004.

[16] David M. Blei and John D. Lafferty. Correlated topic models. In *In Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120. MIT Press, 2006.

[17] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation.

Journal of Machine Learning Research, 3(Jan):993–1022, 2003.

[18] Xiao Bo, Xu Qian-fang, Lin Zhi-Qing, Guo Jun, and Li Chun-guang. Credible association rule and its mining algorithm based on maximum clique. 2008.

[19] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[20] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Measuring semantic similarity between words using web search engines. volume 7, pages 757–766, 2007.

[21] Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.

[22] Christina Brandt, Thorsten Joachims, Yisong Yue, and Jacob Bank. Dynamic ranked retrieval. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 247–256, New York, NY, USA, 2011. ACM.

[23] Leo Breiman, JH Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. *Intelligence*, pages 1002–1007, 1993.

[24] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 649–658, Washington, DC, USA, 2012. IEEE Computer Society.

[25] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, pages 9–16, April 2006.

[26] Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, 2005.

[27] Doina Caragea, Dianne Cook, and Vasant G Honavar. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 251–256. ACM, 2001.

[28] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM.

[29] Joe Carthy and Michael Sherwood-Smith. Lexical chains for topic tracking. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 7, pages 5–pp. IEEE, 2002.

[30] Joe Carthy and Alan F Smeaton. The design of a topic tracking system. In *Proceedings of the 22nd Annual Colloquium on Information Retrieval Research*. Citeseer, 2000.

[31] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168. ACM, 2006.

[32] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[33] Hsin-Hsi Chen and Lun-Wei Ku. An nlp & ir approach to topic detection. In *Topic detection and tracking*, pages 243–264. Springer, 2002.

[34] Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781. Citeseer, 2011.

[35] Zhihang Chen, Chengwen Ni, and Yi Lu Murphey. Neural network approaches for text document categorization. In *The 2006 IEEE International Joint Conference*

on Neural Network Proceedings, pages 1054–1060. IEEE, 2006.

[36] Chris Clifton, Robert Cooley, and Jason Rennie. Topcat: data mining for topic identification in a text corpus. *Knowledge and Data Engineering, IEEE Transactions on*, 16(8):949–964, 2004.

[37] Héctor Cordobés, Antonio Fernández Anta, Luis F Chiroque, Fernando Pérez García, Teófilo Redondo, and Agustín Santos. Graph-based techniques for topic classification of tweets in spanish. *IJIMAI*, 2(5):32–38, 2014.

[38] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[39] Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, pages 1–17, 2007.

[40] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[41] Jianyong Dai and Jianlin Cheng. Hmmeditor: A visual editing tool for profile hidden markov model. *BMC genomics*, 9(Suppl 1):S8, 2008.

[42] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 27. ACM, 2004.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[44] Ludovic Denoyer and Patrick Gallinari. The wikipedia xml corpus. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 12–19. 2006.

[45] Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012.

[46] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A Tomlin, et al. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM, 2003.

[47] Dmoz. Open directory project, `http://www.dmoz.org/`, 2002.

[48] Avinava Dubey, Soumen Chakrabarti, and Chiranjib Bhattacharyya. Diversity in ranking via resistive graph centers. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 78–86, New York, NY, USA, 2011. ACM.

[49] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 2000.

[50] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[51] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, July 2011.

[52] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer, 2008.

[53] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010.

[54] Rafael Ferreira, Rafael Dueire Lins, Fred Freitas, Steven J Simske, and Marcelo Riss. A new sentence similarity assessment measure based on a three-layer sentence representation. In *Proceedings of the 2014 ACM symposium on Document engineering*, pages 25–34. ACM, 2014.

[55] Brendan Frey and Delbert Dueck. Mixture modeling by affinity propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–386. MIT Press, Cambridge, MA, 2006.

[56] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[57] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[58] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. *Knowledge Discovery in Databases: PKDD 2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20-24, 2004. Proceedings*, chapter HIClass: Hyper-interactive text classification by interactive supervision of document and term labels, pages 546–548. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[59] Siddharth Gopal and Yiming Yang. Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322. ACM, 2010.

[60] Allan D Gordon. A review of hierarchical classification. *Journal of the Royal Statistical Society. Series A (General)*, pages 119–137, 1987.

[61] Derek Greene. Matrix factorization for fopic models. *Lecture notes (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.702.4867\&rep=rep1\&type=pdf)*.

[62] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Burgle, Holger Duwiger, and Ulrich Scheel. Faceted wikipedia search. In Witold Abramowicz and Robert Tolksdorf, editors, *Business Information Systems*, volume 47 of *Lecture Notes in Business Information Processing*, pages 1–11. Springer Berlin Heidelberg, 2010.

[63] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, April 2006.

[64] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[65] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674, 2006.

[66] Anna Huang, David Milne, Eibe Frank, and Ian H Witten. Clustering documents using a wikipedia-based concept representation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 628–636. Springer, 2009.

[67] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 465–474. ACM, 2013.

[68] Rishabh K Iyer, Stefanie Jegelka, and Jeff A Bilmes. Fast semidifferential-based submodular function optimization. In *International Conference on Machine Learning (ICML '13)*, pages 855–863, 2013.

[69] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389. ACM, 2008.

[70] Thorsten Joachims. Estimating the generalization performance of a svm efficiently. Technical report, Universität Dortmund, 2000.

[71] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[72] Gordon V Kass. An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, pages 119–127, 1980.

[73] Saurabh S Kataria, Krishnan S Kumar, Rajeev R Rastogi, Prithviraj Sen, and Srinivasan H Sengamedu. Entity disambiguation with hierarchical topic models.

In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1037–1045. ACM, 2011.

[74] James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[75] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[76] Tom Kenter and Maarten de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM, 2015.

[77] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in statistical machine translation. In *Proceedings of EMNLP*, 2014.

[78] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. *CoRR*, abs/1207.1394, 2012.

[79] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119. ACM, 2001.

[80] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273, 2015.

[81] Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1536–1545. Association for Computational Linguistics, 2011.

[82] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[83] David D Lewis and William A Gale. A sequential algorithm for training text clas-

sifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

[84] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397, 2004.

[85] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[86] H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 479–490. AUAI Press, 2012.

[87] Hui Lin. *Submodularity in natural language processing: Algorithms and applications*. PhD thesis, 2012.

[88] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics, 2010.

[89] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[90] Mihai C. Lintean and Vasile Rus. Measuring semantic similarity in short texts through greedy pairing and word semantics. In *FLAIRS Conference*. AAAI Press, 2012.

[91] Hao Ma, Michael R. Lyu, and Irwin King. Diversifying query suggestion results. In *AAAI*. AAAI Press, 2010.

[92] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction*

to formation retrieval. Cambridge university press Cambridge, 2008.

[93] Simone Marinai, Marco Gori, and Giovanni Soda. Artificial neural networks for document analysis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):23–35, 2005.

[94] Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[95] Olena Medelyan, Ian H Witten, and David Milne. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*, volume 1, pages 19–24, 2008.

[96] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 490–499. ACM, 2007.

[97] Prem Melville and Vikas Sindhwani. Active dual supervision: Reducing the cost of annotating examples and features. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 49–57. Association for Computational Linguistics, 2009.

[98] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM, 2007.

[99] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[100] George A Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[101] David Milne. An open-source toolkit for mining wikipedia. In *In Proc. New Zealand Computer Science Research Student Conf*, 2009.

[102] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[103] Ramesh Nallapati. Semantic language models for topic detection and tracking. In

Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3, pages 1–6. Association for Computational Linguistics, 2003.

[104] Mukund Narasimhan and Jeff Bilmes. Pac-learning bounded tree-width graphical models. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 410–417. AUAI Press, 2004.

[105] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[106] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[107] Adler J Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. Hierarchically supervised latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2609–2617, 2011.

[108] Georgios Petkos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Two-level message clustering for topic detection in twitter. In *SNOW-DC@ WWW*, pages 49–56, 2014.

[109] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100. ACM, 2008.

[110] Michael JD Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

[111] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[112] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[113] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[114] Karthik Raman, Thorsten Joachims, and Pannaga Shivaswamy. Structured learning of two-level dynamic rankings. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 291–296, New York, NY, USA, 2011. ACM.

[115] Reuters-21578. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

[116] L Rolling. Indexing consistency, quality and efficiency. *Information Processing & Management*, 17(2):69–76, 1981.

[117] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *Machine Learning: ECML 2006*, pages 413–424. Springer, 2006.

[118] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626, 2006.

[119] Yu A Rozanov. Markov random fields. In *Markov Random Fields*, pages 55–102. Springer, 1982.

[120] Vasile Rus, Nobal Niraula, and Rajendra Banjade. Similarity measures based on latent dirichlet allocation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 459–470. 2013.

[121] Mehran Sahami and Timothy D Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, pages 377–386. AcM, 2006.

[122] Peter Schönhofen. Identifying document topics using the wikipedia category network. *Web Intelligence and Agent Systems: An International Journal*, 7(2):195–207,

2009.

[123] Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge university press Cambridge, 2008.

[124] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, 2000.

[125] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[126] M Mahdi Shafiei and Evangelos E Milios. Latent dirichlet co-clustering. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 542–551. IEEE, 2006.

[127] Carlos N Silla Jr and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[128] Anton Slutsky, Xiaohua Hu, and Yuan An. Tree labeled lda: A hierarchical model for web summaries. In *Big Data, 2013 IEEE International Conference on*, pages 134–140. IEEE, 2013.

[129] Yang Song, Baojun Qiu, and Umer Farooq. Hierarchical tag visualization and application for tag recommendations. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1331–1340. ACM, 2011.

[130] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[131] M. Steyvers and T. Griffiths. Probabilistic topic models. In *Latent Semantic Analysis: A Road to Meaning*, chapter Probabilistic topic models. Laurence Erlbaum, 2007.

[132] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of the tenth*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 306–315. ACM, 2004.

[133] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[134] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 521–528. IEEE, 2001.

[135] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[136] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.*, 40(6):1715–1737, December 2011.

[137] Ashwin Swaminathan, Cherian V. Mathew, and Darko Kirovski. Essential pages. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 173–182, Washington, DC, USA, 2009. IEEE Computer Society.

[138] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S Tan. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1283–1292. ACM, 2009.

[139] Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative markov networks. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 102. ACM, 2004.

[140] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[141] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66,

2002.

[142] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1413–1421, 2014.

[143] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[144] Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 721–728, 2002.

[145] Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. Classification of short texts by deploying topical annotations. In *European Conference on Information Retrieval*, pages 376–387. Springer, 2012.

[146] Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.

[147] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H Witten. Interactive machine learning: Letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.

[148] Christian Wartena and Rogier Brussee. Topic detection by clustering keywords. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 54–58. IEEE, 2008.

[149] Kai Wei, Rishabh K Iyer, and Jeff A Bilmes. Fast multi-stage submodular maximization. In *International Conference on Machine Learning (ICML)*, pages 1494–1502, 2014.

[150] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[151] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2006.

[152] Lihua Wu, Bo Xiao, Zhiqing Lin, and Yueming Lu. A practical approach to topic detection based on credible association rule mining. In *Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on*, pages 227–231. IEEE, 2012.

[153] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. Short text clustering via convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 62–69, 2015.

[154] Semih Yagcioglu, Erkut Erdem, Aykut Erdem, and Ruket Cakıcı. A distributed representation based query expansion approach for image captioning. In *Annual Meeting of the Association for Computational Linguistics*, 2015.

[155] Lian Yan, Robert H. Dodier, Michael Mozer, and Richard H. Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *International Conference on Machine Learning (ICML)*, pages 848–855, 2003.

[156] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

[157] Wen-Tau Yih and Christopher Meek. Improving similarity measures for short segments of text. In *AAAI*, volume 7, pages 1489–1494, 2007.

[158] Yisong Yue and Thorsten Joachims. Predicting diverse subsets using structural svms. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1224–1231, New York, NY, USA, 2008. ACM.

[159] Torsten Zesch and Iryna Gurevych. Analysis of the wikipedia category graph for nlp applications. In *Proceedings of the TextGraphs-2 Workshop (NAACL-HLT 2007)*, pages 1–8, 2007.

[160] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[161] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, 10(2):141–168, 2005.

[162] Jun Zhu, Amr Ahmed, and Eric P Xing. Medlda: Maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 1257–1264. ACM, 2009.

[163] Xiaojin Zhu. Semi-supervised learning literature survey, `http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html`, 2008.

[164] Guowei Zu, Wataru Ohyama, Tetsushi Wakabayashi, and Fumitaka Kimura. Accuracy improvement of automatic text classification based on feature transformation. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 118–120. ACM, 2003.

[165] Sven Meyer zu Eissen and Benno Stein. Analysis of clustering algorithms for web-based search. In *Practical Aspects of Knowledge Management*, pages 168–178. Springer, 2002.

[166] Arkaitz Zubiaga and Heng Ji. Harnessing web page directories for large-scale classification of tweets. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 225–226. ACM, 2013.